Managing Bandwidth and Traffic via Bundling and Filtration in Large-Scale Distributed Simulations

THESIS

Roberto C. Sánchez, Second Lieutenant, USAF

AFIT/GCE/ENG/06-06

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/GCE/ENG/06-06

Managing Bandwidth and Traffic via Bundling and Filtration in Large-Scale Distributed Simulations

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Computer Engineering

Roberto C. Sánchez, A.A.S., B.S.Cp.E.

Second Lieutenant, USAF

June 2006

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AFIT/GCE/ENG/06-06

Managing Bandwidth and Traffic via Bundling and Filtration in Large-Scale Distributed Simulations

Roberto C. Sánchez, A.A.S., B.S.Cp.E.
Second Lieutenant, USAF

Approved:

/signed/	06 June 2006
Dr. Kenneth Hopkinson (Chairman)	date
/signed/	06 June 2006
Dr. Barry E. Mullins (Member)	date
/signed/	06 June 2006
Maj Scott R. Graham, PhD (Member)	date

Abstract

Research has shown that bandwidth can be a limiting factor in the performance of distributed simulations. The Air Force's Distributed Mission Operations Center (DMOC) periodically hosts one of the largest distributed simulation events in the world. The engineers at the DMOC have dealt with the difficult problem of limited bandwidth by implementing application level filters that process all DIS PDUs between the various networks connected to the exercise. This thesis examines their implemented filter and proposes: adaptive range-based filtering and bundling together of PDUs. The goals are to reduce the number of PDUs passed by the adaptive filter and to reduce network overhead and the total amount of data transferred by maximizing packet size up to the MTU. The proposed changes were implemented and logged data from previous events were used on a test network in order to measure the improvement from the base filter to the improved filter. The results showed that the adaptive range based filter was effective, though minimally so, and that the PDU bundling resulted in a reduction of 17% to 20% of the total traffic transmitted across the network.

Table of Contents

List of Figures

List of Tables

Managing Bandwidth and Traffic via Bundling and Filtration in Large-Scale Distributed Simulations

I. Introduction and Problem Definition

1.1 Overview

Distributed simulation systems make possible a wide variety of training and operations functions. For example, it is possible for two pilots located at two geographically separated locations to fly together against threats simulated at yet a third location. All of this occurs without anyone leaving home station and without the need to fly costly and dangerous sorties. This arrangement is possible because protocols, such as the Distributed Interactive Simulation (DIS) protocol, allow entities connected via a network to exchange packets. These exchanged packets form the basis of the distributed simulation environment.

The distributed simulation environment is suitable for all manner of training and, recently, operations. For example, two pilots can fly a simulated dog fight. Two tank battalions can face off on the battlefield of a place that only exists in the simulated world. Simulated unmanned aerial vehicles (UAVs) can patrol the simulated skies over the battlefield where the tanks are facing off. In fact, the possibilities are limited only by the imagination of the simulation designers and the capacity of networks which support the simulated environment. While there are numerous benefits, there are also many factors, including realism, latency and bandwidth, that can make a distributed simulation less than optimal.

While realism is important, it does not need to be perfect. In fact, pilots flying in simulators and in real sorties have exhibited some of the same psychological and

physiological responses [21], indicating that a less realistic simulation is still effective in evoking the desired responses. Latency is largely a function of the underlying network infrastructure. While the underlying network infrastructure also affects bandwidth, there is much that can be done at the application and protocol levels to minimize and manage the consumption of bandwidth by the simulation traffic.

The need to understand and manage bandwidth consumption by simulation traffic is seen by extending the previous example to include 400 pilots at 20 different bases, or even 4000 pilots at 200 different bases. While the last example may seem contrived, as the likelihood of having 4000 pilots from 200 different bases simultaneously involved in a distributed simulation is low, a VIRTUAL FLAG exercise held at the Distributed Mission Operations Center (DMOC), Kirtland AFB, NM, in April 2005 included connections to at least 30 remote sites (with many sites connecting multiple nodes) across the country [23].

Given the relative capacity of networks today, especially local area networks that are connected via Gigabit Ethernet or better, many simulations still have room to grow. However, in many cases, networks at different sites are connected together via leased lines, or even over the public Internet, with significantly less bandwidth. These long haul link connections generally limit the amount of traffic that can be exchanged between the connected networks. This makes it important to carefully study how the current bandwidth is being used and ways that its use can be managed. This would allow a more precise knowledge of the utilization of current bandwidth resources. Understanding bandwidth utilization now will also increase the capability of the networks with respect to the ability to scale to meet future bandwidth needs.

Engineers who managed the network during the April 2005 VIRTUAL FLAG exercise have stated that bandwidth is not a great concern because utilization rarely approached capacity of the lowest capacity links (T1 lines) [23]. (It is important to note, however, that the DMOC already implements a filtering system in order to manage bandwidth.) Most remote sites were connected to the DMOC via dedi-

cated T1 links. The capacity of a T1 is 1.544 Mbps, and the links for this particular exercise rarely exceeded 1200 kbps bandwidth utilization [23]. This is a normal occurrence for most exercises conducted at the DMOC. The limiting factor encountered in nearly all exercises by the DMOC is that the number of entities (i.e., live, virtual and constructive) involved in any one simulation can easily overwhelm simulator on the network. Even within their own fast local area network, the DMOC engineers implement filtering mechanisms to prevent large amounts of traffic from crashing or otherwise degrading the various simulators [23].

1.2 Basics of DIS Simulations

In general, DIS simulations are composed of a number of entities connected via a network. Those entities might be live (real people in real equipment that is connected to the simulated environment via remote links), virtual (real people in simulated equipment) or constructive (simulated people in simulated equipment). In all cases, the connection to the simulated environment is made by the exchange of packets, called protocol data units (PDUs) in DIS simulations. In the case of only virtual and constructive entities, it is possible to have all entities on the same small local network. However, the DIS protocol is designed to operate over large and geographically separated networks. Thus, it is possible that the people interacting via the simulated environment are in fact located in different parts of the country or even the world.

The exchange of PDU data follows a very specific pattern. PDUs are one of a number of different types, as specified in [12]. Each has a specific size and carries specific data. In this way, each entity has a set of possible that it can send to communicate its state and activities to other entities in the simulated environment. In the same way, each entity can expect that will receive PDUs only of the type specified in the standard. This regimented approach ensures that any simulator designed to meet the DIS standard is able to communicate with any other simulator that meets the same standard. This is similar to how client-server (e.g., HTTP, SMTP) and peer-to-

peer (e.g., BitTorrent) protocols work. As long as the various clients, servers or peers conform to the standard, it does not matter when or how development occurred, as they are capable of communicating with other clients, servers or peers that can speak the same protocol.

For example, in a particular DIS simulation, there might be a strike fighter and a patrolling tank. Each entity is aware of the other based only what PDUs it broadcasts into the simulated environment. Each entity is responsible for broadcasting entity state (ES) PDUs periodically to ensure that its location and movements can be tracked by other nearby entities. To continue the example, as the fighter approaches the tank it will send out entity state PDUs detailing its location, speed, altitude and other data about its state. The tank will do the same. Once the tank sights the fighter, it will take evasive action and continue to update its state with its new movements. Once the fighter reaches the appropriate distance, it will fire. When the fighter fires a missile on the tank, it will broadcast a fire PDU which will contain details of the type of weapon fired and the speed and direction. The fighter may also radio to its controller, which requires sending a signal PDU. The fire PDU allows the tank simulator to visually represent the event so that its occupants can see it. Once the fighter's missile hits, the fighter will broadcast a detonation PDU. The detonation PDU will detail the location, type and force of the blast. The tank will receive that information and take the appropriate damage and subsequently update the ES PDUs it is broadcasting.

The above example is simplistic, but it illustrates the most basic functions in a DIS simulation. There are several dozen PDU types to meet a wide variety of requirements. However, entity state and signal PDUs remain the most common. As the simulation scales, the number of PDUs being broadcast grows quickly. However, because all traffic is broadcast, it can be received by all entities and ensures that each has a complete representation of the simulated environment.

1.3 Type of Research

This thesis presents applied research in the area of large scale distributed simulations. The problem in this case involves improving an existing Distributed Interactive Simulation (DIS) application-level packet filter to provide a more advanced, fine-grained filtering capability, and a more efficient use of available packet payload, resulting in a reduction in the network traffic. The proposed methods were applied as modifications to an existing and currently deployed filter implementation. Success was measured by comparing the number of "relevant" packets allowed to pass by the filter and also by the amount of reduction in overhead and total traffic during a simulation. That is, the filter processes packets (specifically, DIS PDUs) and chooses to forward them based on whether or not the specified filtering rules lead to the conclusion that the particular packet being examined is relevant. Some irrelevant packets are allowed through because the filter is too coarse. If the improved filter is able to allow a smaller subset of packets without adversely affecting performance, then success has been achieved. The reduction in the amount of traffic being sent is the measure of the magnitude of success.

1.4 Technical Area

Distributed simulations are a specific type of distributed system. Features like reliability, scalability and cost effectiveness have driven the adoption of a distributed approach to computer modeling and simulation. Distributed systems attempt to provide users with access to a capability that may not otherwise be available because of geographical separation, cost, etc. In the case of distributed simulations, they also make it possible to train personnel in situations that would be dangerous or deadly in real life. There is a great deal of ongoing research in the area of distributed systems.

1.5 General Problem

As with any complex system, distributed simulations face a myriad of limitations and problems which must be overcome. Some researchers contend that bandwidth is

a major limitation and propose solutions which trade processor cycles to save network packets However, many proposed solutions are difficult or impossible to implement in a real simulation. In fact, the increasing speed of modern networks makes it seem as though there may come a time when even bandwidth hungry applications will have all the bandwidth they need, within reason. A related, though slightly different, problem is traffic management. This includes bandwidth-saving techniques and filtering of packets to reduce or eliminate unnecessary traffic, but also includes higher level routing decisions and modification of packets in flight. This last area has not been as intensely studied.

1.6 Specific Problem

Effectively managing simulation traffic requires consideration of a number of issues. For example, the topology of the network on which the simulation will be executed must be carefully designed and managed. In particular, many simulator systems generate large amounts of traffic and a poorly designed network can quickly become a bottleneck. Also of concern are cases where the simulation network is an internetwork composed of a number of networks managed by different organizations. Add to that instances where the simulation network is not dedicated and must also support general purpose use. Both of these situations necessitate working with already available resources and generally make optimizations difficult or impossible.

Further, the actual design of the individual simulator systems also has an effect on the amount of traffic generated. However, it can be difficult to exert control over the design of individual simulator systems, which can originate from third-party organizations or from commercial-off-the-shelf (COTS) sources. Further, the protocol to which the simulator conforms dictates certain parameters. For example, DIS simulators generate protocol data units (PDUs) conforming to the DIS standard. These PDUs tend to be small and, as a result, wasteful in terms of the overhead required to send. In addition to the amount of traffic generated, each simulator's design inherently places a limit on the amount of possible simulation traffic that it can handle.

Both cases, of the network architecture and the individual simulator limitations, are usually beyond the control of the designers of the large-scale simulation events.

When confronted with a situation where it is desirable to provide an optimal environment for a large distributed simulation, yet the simulations and underlying networks cannot be extensively modified, the best approach is to filter the traffic as much as possible [23]. This allows the efforts of network engineers to be focused on overcoming the design limitations of most simulators rather than bandwidth limitations, which tend not to be as much of a problem; simulators can fail if left exposed to the unfiltered amount of traffic generated on the LAN during an exercise [23]. Filtering, rather than the actual bandwidth itself, is the main problem with regard to the traffic generated by the various simulators [23]. That said, bandwidth can still become an issue over the slower links used to connect distant sites to the simulation network. This issue will be discussed in detail later.

The problem then becomes one of developing an optimal filtering algorithm and strategy such that only the minimum amount of traffic that is relevant to the entities on a particular part of the network pass the filter. This means that more than one filter is needed separating each site or network segment which is to have its traffic filtered. In other words, there is a filter on every link between networks, local and remote. The DMOC engineers have already deployed a solution to this particular problem. However, their solution is born of necessity and is based on the personal experience of their engineers and working knowledge of their own particular configuration. There is little or no theoretical background to their work. The purpose of the work presented in this thesis is to make improvements to their existing solution.

1.7 Hypothesis

By modifying the existing range-based filtering algorithm to adapt the range during the simulation, it is possible to reduce the number of PDUs being passed by the filter. Additionally, by bundling together PDUs, it is possible to significantly reduce the network overhead expended by sending many small individual packets.

1.8 Contributions

- Describe a pre-existing implementation of an effective filtration algorithm currently in use in some of the largest distributed simulation events in the world
- Explore the effects of using an adaptive range-based filtering approach, instead of the current static range-based filter
- Explore the effects of bundling PDUs in order to reduce the overhead and total network traffic of the simulation
- Implement the proposed modifications and test the updated filter for improvement

1.9 Attributes

1.9.1 Novelty. Much current research, in the areas of bandwidth reduction and interest management, advocates techniques like compression of packets or modification of existing simulation systems or the underlying infrastructure. The research presented in this thesis is novel in the sense that it explores a technique for reducing bandwidth and managing traffic by filtering the traffic without altering the underlying network or modifying the existing simulation systems.

1.9.2 Significance. Since previous academic work has been concerned with approaches that require modifications that are difficult to scale in a heterogeneous environment, this research will contribute to advance the state-of-the-art by studying a method for reducing bandwidth and managing interest which can be implemented with minimal disruption to existing systems. That is, while some modifications to the network infrastructure are inevitable, such as adding machines for filtering, this filtering is accomplished without modification to the existing simulators themselves.

1.9.3 Usefulness. While much research already exists in the areas of bandwidth reduction and interest management, many proposed approaches are not particularly useful in the real world. The research presented in this thesis aims to present

a truly useful approach, which can be applied equally well in research systems and in real world production systems. This is a lofty goal, to be sure, but helps to keep in perspective the ultimate objective of performing relevant research.

1.10 Summary

There currently exists a solid background and foundation of work in distributed systems. The field is ripe with opportunities for relevant and useful research. The research presented in this thesis builds on this existing foundation to develop an advancement to the state of the art and an improvement to an existing system that is in production use in some of the largest distributed simulation events in the world.

1.11 Road Map

This section briefly describes the layout of the rest of the thesis.

Literature Review Chapter II is a discussion that will establish the relevant background material and state-of-the-art in distributed simulation.

Current System Configuration and Deployment Chapter III examines the preexistent network topology and infrastructure of the DMOC by discussing the preexisting algorithms and filtering strategies used on their networks in support of large-scale distributed simulation events, such as VIRTUAL FLAG exercises. This forms the baseline system against which the research in this thesis is measured to determine the level of success.

Proposed Changes and Experimental Design Overview Chapter IV discusses overall approach of the design and the associated criteria for a good solution, along with what external issues were considered. A high level overview of the experimental evaluation criteria is also given.

Detailed Experimental Design and Results A key aspect of this research was to design test cases that allow a fair comparison between the new implementation

and the existing implementation of the DIS Filter software. Chapter V discusses the test cases and the results that were collected.

Conclusions Chapter VI provides an analysis of the data and draws the relevant conclusions, in addition to providing recommendations for future work.

II. Literature Review

2.1 Background

2.1.1 Foundations of Distributed Simulations. Modeling behaviors of physical systems has been done from the earliest days of science. Computer modeling and simulation has served to increase the scale and complexity of the systems which can be modeled and the speed with which the models can be simulated. In fact, the very first digital computer, ENIAC, was developed for the U.S. Army to calculate artillery trajectories. The formulae used to compute the trajectories are models of ballistic flight.

Prior to the advent of digital computers, simulations were almost exclusively mathematical and were laboriously hand computed. As with many other fields of math, science and engineering, the need to work every problem by hand greatly limited how much could be accomplished and how accurately. Modern computers effectively lifted that barrier because a digital machine is capable of repeatedly performing many mathematical operations without tiring or making a mistake. After the arrival of the digital computer, systems intended for modeling and simulation were built to provide as much computing power as possible. This made the systems extremely large and expensive.

The dawn of the information age and the arrival of the personal computer revolution changed the way in which computing power was viewed. Parallel computing was extended to harness the largely untapped computational power of large numbers of workstations that could be inexpensively assembled together. Applying the idea of distributed simulations to such systems posed a unique set of challenges. With a simulation running entirely on a single computer, all of the information is always locally available. On the other hand, running a simulation in a distributed manner to take advantage of greater available computational power introduces a great amount of additional complexity.

Fundamentally, the benefits of distributed simulation are reduced execution time, geographical distribution, heterogeneity of hardware, and fault tolerance [8].

There is solid groundwork and discussion of the variety of approaches to distributed simulations and how to deal with the challenges that must be overcome to realize those benefits; Fujimoto details a wide variety of associated issues [8].

2.1.2 Timekeeping and Causality. The concept of timekeeping is central to the execution of distributed simulations. With a simulation that executes entirely on a single computer, every entity references the same clock. In a distributed simulation, each computer provides a clock to the entities which execute locally, but the clocks are not always synchronized. In addition, the entities must consider wallclock time, physical time, simulation time and various other times relating to the optimism and/or lookahead of the simulation. For example, the entity must know whether or not events can occur in its past and, if so, how far back such events can occur [8].

Even more important than timekeeping is causality, which ensures that the principle of cause and effect is never violated. There are essentially two ways to ensure that causality is maintained: conservative simulation or optimistic simulation with rollback [8]. Conservative simulations make the assumption that a particular event can only be processed once it is impossible for any events to arrive from other entities that could possibly be scheduled prior. Such events are considered "safe." The advantage to the conservative approach is that once an event is processed its result is fixed and cannot be undone. The disadvantage is that it is susceptible to deadlock and suffers from poor performance because many entities waste time waiting.

On the other hand, optimistic simulations assume that every event is "safe" and if it turns out that the assumption was wrong, the entity rolls back to a state prior to the first incorrectly processed event. Optimistic simulation introduces a whole host of additional concerns. For example, if an entity executes events that occur at simulation times $t = 15$, $t = 20$ and $t = 25$ and afterward an event arrives with time stamp $t = 10$, the entity must undo, or "rollback," the already processed events. The rollback of those events can in turn cause further rollbacks, called secondary

rollbacks [8], which can be numerous. The concept of lookahead can be employed to ensure that events cannot happen before a certain time in the past.

The entity (in optimistic simulation) must remember what messages have already been sent so that it knows which to recall should the need arise. The entity must must also know when it is safe to "commit" irrevocable actions, such as writing to disk [8]. Additionally, memory management becomes very important. Maintaining extensive queues of previously processed messages can become impossible given available memory. Fujimoto describes different memory management schemes that specifically seek to optimize the number of messages retained [8] and also discusses a number of other issues related to optimistic simulations [8].

2.1.3 Available Simulation Technologies. When the need exists to implement a distributed simulation, the developers must choose from a dizzying array of available techniques, technologies and frameworks. The potential use of High Level Architecture (HLA), Common Object Request Broker Architecture (CORBA) and Remote Method Invocation (RMI) in distributed simulations has been examined [4]. Essentially, the developers of a simulation must carefully consider a number of factors, including support for legacy systems, programming language flexibility, security and timekeeping. The specific targets of the available technologies are also important to developers. CORBA and RMI are general purpose and HLA was designed with distributed simulations in mind [4]. Essentially, the developers must decide if the "mandated" and "guaranteed" interoperability of HLA is worth the additional overhead of developing within HLA. If a simulation is being designed to interact with other HLA-compliant systems, the use of HLA is a given. For simulations not bound by that particular requirement, more choices exist.

Another technology, which was not examined in [4], is Distributed Interactive Simulation (DIS). While DIS and HLA are both targeted at distributed simulations, they take fundamentally different approaches. The fundamental difference between DIS and HLA is that DIS employs a *broadcast* approach, while HLA employs a *publish-*

subscribe approach [16], [7], [12], [13]. That distinction alone can account for a large difference in performance when it comes to bandwidth utilization. However, the DIS model should not be rejected, because there is not one simulation that can meet all possible requirements for everyone [7]. Regardless, it is necessary to point out that nearly all distributed simulations take either a DIS-type approach or an HLA-type approach to moving information around. Both approaches can be optimized and each has benefits and limitations. It is also important to note that because HLA is a standard mandated for use within the DoD, there is a great deal of commercial and academic focus on HLA over DIS.

Regardless of the general emphasis on HLA over DIS in industry and academia, engineers at the DMOC have stated that the simulations which they oversee benefit from the very precise specification of DIS packets available in IEEE Std. 1278 [23]. That is, although HLA has interoperability as a prime feature, it is also very flexible. On the other hand, DIS is completely inflexible in the format of the packets, also called protocol data units (PDUs). Experience at the DMOC has shown that the rigid framework of pre-specified DIS PDUs is far more readily interoperable than the HLA Object Model Template (OMT) approach [23].

2.1.3.1 Developing HLA Federates. A number of researchers have examined various aspects of developing federates for use in HLA simulations. While these activities are particular to HLA simulations, many of the observations are more widely applicable. For example, using the Defense Modeling and Simulation Office (DMSO) Federation Development and Execution Process (FEDEP) as a guide for developing a federate in a NATO environment revealed that the FEDEP likely leaves some important points uncovered with regard to interoperability [3]. The same can probably be said of any guideline or standard that gains widespread acceptance and use.

When an available technology is chosen and its framework is used to develop a simulation, any limitations it has may or may not become an issue. Though, if

limitations pose a problem, extending the framework can usually overcome those limitations. For example, the DMSO Run-Time Infrastructure (RTI), version 1.3-NG, was extended to provide additional capabilities for application development and data interchange [10]. This allows the modified version to interface with Modular Semi-Automated Forces (ModSAF) and One Semi-Automated Forces (OneSAF) terrain databases. This extensibility is key to maximizing the benefit of working within a predefined framework.

There can be difficulty in developing "parallel time-constrained federates within an HLA" [22]. A case study of building time-constrained HLA federates with the PARSEC simulation language discovered a number of issues with respect to the performance of the RTI, the architecture of the simulation and the definitions of time within the simulation [22]. In this case, it became apparent that while the HLA is meant to be general purpose, it is not well suited for all possible types of federates.

The key aspect to developing a federate is to develop a sound conceptual model so that the experiment designer and federate designer can each focus on their key parts [33]. In this case, the development was of a federate targeted specifically for experimentation. As with [3], development of this federate also followed the FEDEP. It is worth noting that the FEDEP itself bears a striking resemblance to the outdated waterfall software development model in the sense that it assumes that each step will be completed in its entirety before the next step proceeds. Regardless, that the FEDEP is still useful and that it need not conform to this linearity—it can be executed in a cyclic or concurrent fashion [33].

2.1.3.2 Cloning HLA Simulations. HLA can be optimized by cloning HLA federates in a simulation, thereby eliminating repeated computations [5]. For example, if two branches of a particular decision are to be explored, it is better to clone the involved federates and execute both branches in parallel. Since not all such branching requires that every federate be cloned, incremental cloning can be used only when necessary and Data Distribution Management (DDM) can be used to

ensure messages are routed correctly when federates are participating in more than one simulation at a time [5].

The steps of the cloning process can be enumerated [5]:

1. Initial cloning
2. Synchronizing federation
3. Handling in-transit events
4. Updating scenario tree
5. Making clones
6. Replicating system states
7. Leaving cloning mode

These steps represent an interesting approach to optimizing HLA simulations for performance. The authors state that this approach results in reduced computation. On the surface it seems that the assertion is justified, but there are no figures or data to back up the authors' statement.

2.1.4 Available COTS Simulation Products. The use of commercial off-the-shelf (COTS) products has long been emphasized by the DoD and increasingly within industry as the cost savings over developing custom in house solutions has increased. The area of simulation is no different. A variety of commercial vendors have come forward with a wide variety of products aimed at all levels and complexities of simulations. Most notably, software wrappers are necessary since no two COTS simulation packages are alike [1]. However, it is likely that the observation about wrappers is based on the idea that many COTS solutions use the HLA or a similar approach. Practical experience with DIS simulations implies that such wrappers are rarely necessary [23].

2.1.4.1 Openness of COTS Tools. Another important distinction that must be made is between fully open, partly open and fully closed COTS simulation tools. The level of openness of a particular tool directly correlates to the flexibility which it provides. Fully open tools—those in the public domain or under a free

software license, such as the GNU General Public License or the Berkeley Standard Distribution License—provide the greatest flexibility because they can be modified, and the modifications can be redistributed and made part of the tool for others to use. Partially open tools may provide source code but prohibit modification and redistribution, or they may provide additional technical documentation under a non-disclosure agreement. Fully closed tools provide the least flexibility and also the greatest simplicity. If a closed tool meets the requirements of a particular development effort, then the added flexibility is likely not necessary.

2.1.4.2 MODSIM III. The MODSIM III simulation language is a COTS tool where the network communication is slow and unreliable; the tool should do more in this area to help the simulation developer [14]. Though Johnson extols the benefits of a simulation language over the direct use of C and C++ and even provides a case study which addresses his concerns, there are no experimental results to validate any benefits resulting from using this tool.

2.1.4.3 JTLS and JCATS. An effort to federate two existing commercial simulation tools, the Joint Theater Level Simulation (JTLS) and Joint Conflict and Tactical Simulation (JCATS) was undertaken and number of technical issues, including the prior planning that had to take place with respect to the publication of information about the entities and the management of objects at runtime [2]. In order to facilitate multi-resolution modeling (MRM), attributes of objects are shared [2]. Only the attributes which are of interest to a particular federate are ever transferred. Those which are of no interest or which cannot be utilized at all by a particular federate are not transferred to that federate [2]. HLA's time management was beneficial to the implementation effort [2]. This is more an example of composing COTS products, as opposed to using COTS products to develop, but it illustrates nicely the range of roles that COTS products can fill in the distributed simulation area.

2.1.4.4 SLX. Straßburger, Schulze, Klein and Henriksen cite numerous reasons why C++ is not well suited for developing simulations—difficult to learn, ease of making mistakes, no built-in mechanism for describing parallelism and debugging tools which lack awareness of the simulation environment [24]. However, the points made about requirements for programming to the HLA interface seem to support the use of C++, as many of the requirements lend themselves to C++ programmers or easily accomplished in C++. Regardless, they chose the Simulation Language with Extensibility (SLX) COTS tool (developed by Henriksen's Wolverine Software Corporation) to develop an Internet-based simulation.

Straßburger, et al., explain four ways in which the integration of existing tools and HLA can be accomplished [24]:

- "Re-implementation of the tool with HLA-extensions."
- "Extension of the intermediate code."
- "Usage of an external programming interface."
- "Coupling via a gateway program."

They chose a wrapper approach, similar to a gateway program, and observed acceptable performance.

2.2 State of the Art

2.2.1 Large-Scale Simulations. An examination of large-scale distributed simulations with HLA has shown that limited bandwidth is a bottleneck [19]. Bandwidth is, in fact, one of the bottlenecks for a large simulation with completely unfiltered traffic. In this case, a three-level control mechanism (3LCM) and a dynamic filtering strategy (DFS) that minimizes the overlap of update regions is proposed [19]. Whenever the publishing region of one or more federates overlaps with the subscription region of one or more federates, the object attributes for the overlapping region are routed to the subscribing federate. By minimizing the overlap they expect to see less traffic being routed.

The DFS has the following features [19]:

- Uses Java RMI to have the federates communicate with the RTI console.
- Sacrifices some information consistency (e.g., a lower fidelity of data is used) in order to maintain smooth interactions and reduce the update frequency.
- Uses a shared repository to ensure absolute consistency among the various entities at the first level of the 3LCM.
- Uses "blind broadcasting" (a form of interest management) at the second level of the 3LCM.
- Uses a predictive simulation scheme (a second order polynomial model that resembles a dead reckoning approach) at the third level.

In addition to trying to minimize the actual overlap, the DFS tries to minimize the occurrence of redundant updates by using a sophisticated algorithm that can determine the efficiency of the filtering and then decide whether the region can be moved or resized and still maintain the required efficiency. At the predefined threshold points, which correspond to the points at which continuing with the same scheme will result in greater bandwidth utilization than switching, the three-level control mechanism (3LCM) transitions from the initial shared repository to the information interest management and then to the predictive simulation scheme [19]. In each of the three modes of operation, the 3LCM maintains performance equal to the particular model it was using at each stage. This means that as the number of federates increases, performance of the server stays reasonably constant until the number of clients grows very large [19].

An experiment was carried out, which compared the HLA-based distributed simulation with and without the DFS, showed that the DFS provided a slight improvement [19]. Having multiple federates running a 3LCM provides better performance for larger numbers of overall federates in the simulation. For smaller numbers, having the 3LCM either residing with a single federate (centralized) or multiple federates (distributed) resulted in equal performance [19].

This approach is very promising for a number of reasons. First, the publish-subscribe model is used by other distributed simulation systems. The DFS and 3LCM

can be applied in a more general sense to non-HLA simulations. Second, the variation in approach shows that a generic approach would likely not have worked as effectively. Having the different levels of filtering resulted in at least parity with the original model at each stage [19]. Trying to force one exclusive approach where more than one approach is needed is likely to result in suboptimal performance. Finally, the sophisticated efficiency algorithm developed could be adapted to measure the efficiency of similar approaches on other simulations.

2.2.2 Load Sharing Optimization.

2.2.2.1 Sharing the Load with Multi-Threading. One approach to performance optimization for distributed simulation is to balance or share the load with multi-threading. This has the benefit that a multi-threaded application can automatically take advantage of additional processors on the local machine. A multi-threaded RTI was used to perform load distribution [26]. In addition to the standard benefit of a multi-threaded application, the proposed approach takes advantage of the ability to migrate threads from one host to another.

Since each federate runs in its own thread, it is then also possible to migrate federates from one node to another on the network. One problem when migrating federates is that when using a multi-threaded RTI the callbacks are delivered immediately [26]. This problem is solved by adding a wrapper around each federate that is used to communicate with the load distribution system so that the rest of the communications occur through the RTI, as expected [26]. Since the communications will not break by migrating federates, the knowledge engine can accept input from the load monitor and scenario monitor in order to make decisions as to which federates should migrate and when. The knowledge engine can then actually migrate the federates without incident.

The federate's main simulation loop and the migrate function are converted into reactive programs to allow an interruption of the main loop, for the purpose of

migrating the federate, and then resuming execution at the same state previously held [26]. The federate must rejoin the federation once it has arrived at the target host since the RTI does not understand load distribution and would otherwise not be aware that a federate has simply "moved." A "homeQ," "remoteQ" and "residualQ" and a special region to which "both" federates can subscribe are used to facilitate the migration process and ensure that no messages to the migrating federate are lost [26]. This is accomplished by having two instances of the federate and then combining the messages received by both when possible. Latency for migration of the Publisher federate was 35% to 46% better than Yuan's protocol [32]. Latency for migration of the Subscriber federate was not better, with Yuan's being 6% to 10% better than the one Tan and Kim presented [32]. The utilization can be improved up to 17% by the approach presented in [26].

2.2.2.2 Sharing the Load with Scheduling. In addition to migrating executing threads, it is possible to balance the load of a running simulation with intelligent scheduling. Load sharing was considered in a heterogeneous environment where half of the CPUs run at twice the speed of the other half [15]. The approach divides the jobs into first class jobs for fast CPUs and second class jobs for slow CPUs. The jobs can be migrated either via a sender-initiated mechanism or a receiver-initiated mechanism. The goal is to have no idle CPUs when any CPU is heavily loaded and for the load to be even across all CPUs [15].

The two types of scheduling policies employed are static—simple with no maintenance of state information—and dynamic—more complex with maintenance and evaluation of state information [15]. The dynamic approach exhibits better performance, but at the expense of greater complexity. Jobs can be scheduled probabilistically, deterministically, or adaptively. Probabilistic scheduling ensures dedicated (first class) jobs go randomly to fast CPUs and generic (second class) jobs go randomly to slow CPUs using "state independent branching probabilities" [15]. Deterministic scheduling routes based on system state and can employ one of two policies. The first

deterministic policy sends dedicated jobs into the shortest queue of the fast CPUs and generic jobs into the shortest queue of the slow CPUs [15]. The second deterministic policy sends dedicated jobs into the shortest queue of the fast CPUs and generic jobs to the CPU with the lowest response time (fast or slow) [15]. The adaptive policy can be stated simply as one of moving jobs from heavily loaded slow processors to idle processors in order to keep the load as balanced as possible [15].

Only queued jobs (not executing jobs) can be migrated; otherwise it is too expensive and complex. The overhead of requeuing and then migrating a process that has already begun execution is not worth the overhead. The queues and scheduling were simulated as an open queuing network with $P = 16$ heterogeneous nodes and a high speed network connecting each node. The simulation included one arrival stream for each kind of job (dedicated or generic), with service requirements being the the same for both types of jobs; jobs are independent (they do not need to synchronize with each other). They found that the probabilistic scheduling policies provided the worst overall performance and that the deterministic algorithms provided the best overall performance, considering the overhead of the scheduling algorithm [15].

Overall, the following scheduling policies were considered: probabilistic (Pr), probabilistic with migration (PrM), shortest queue (SQ), shortest queue with migration of generic jobs (SQM), least expected response time for generic jobs-maximum wait for dedicated jobs (LERT-MW), LERT-MW with migration (LERT-MWM). In the end, "SQM is the best method when individual job class performance and fairness is important" [15].

2.2.3 Shared State Optimization. Another potential point of optimization is state information, such as the idea of shared state synchronization and management in distributed HLA simulations [9]. While the approach is specifically targets HLA simulations, the concepts should be equally applicable to any distributed simulation that maintains state information.

Specifically, an algorithm is proposed to replace some timestamp order (TSO) messages with receive order (RO) messages to allow federates to advance time further without constraint [9]. The solution tries to relax the lookahead constraint. This directly deals with one of the issues discussed by Fujimoto, that lookahead must be constrained to ensure that no one entity advances too far ahead of the rest [8]. The constraint helps prevent excessive rollbacks that are triggered by a rollback far in the past of a single entity that has greatly advanced ahead of the other entities in a simulation.

It is noted that shared state can introduce situations which result in sequential execution of federates [9]. Of course, this is no better than if the entire simulation were executing on a single processor. The message proposed replacement allows federates to choose a lookahead value that is greater than zero. To make certain that the causality constraint is not violated, the request-reply approach introduces a history file and the push approach introduces a future file [9]. If either of the lists are empty, the requesting entity must wait so that causality is not violated.

The proposed approach was implemented as middleware layers between the simulation federate and the RTI Ambassador and also between the RTI and the federate ambassador. It was observed that the push scheme (with a future file) performs the best with increasing performance corresponding to increasing lookahead [9]. The pull scheme (with a history file) also performs well. Both outperform the TSO approach.

Like other approaches, relaxing the lookahead constraint requires modification of the run time structure of the simulation. Fortunately, the middleware implementation is more flexible than simply modifying the existing RTI. This also allows the approach to be more easily adapted to other situations and simulations.

2.2.4 Time Warp Scheduling Optimization. Time warp depends on the presence of reliable communications and allows for out of order issue and delivery of messages [8]. Because conservative time warp approaches limit the performance of simulations, aggressive approaches are often necessary. However, an aggressive or

risky time warp results in lots of rollback, with aggressiveness and risk being potential sources for causality errors [6]. The approach was taken of choosing logical processes with low probability of being rolled back and which will cause lower fan-out [6]. To complicate matters further, it is important to consider scheduling of the next LP on the processor for large/complex systems where it is likely that more than one LP is run on each physical processor [6].

Rather than simply execute as many events as quickly as possible, the proposed method takes a more measured approach factoring in fan-out and timestamps [6]. This approach results in the following general treatment of events: low timestamp events are executed sooner to prevent primary rollbacks and large fan-out events are executed later to prevent secondary rollbacks. This illustrates the careful balance that must be struck in order to maximize the effectiveness of an aggressive or risky time warp approach.

The Aggressiveness/Risk Effects based Scheduling (ARES) has the following features [6]:

- modification of Lowest-Timestamp-First algorithm (LTF)
- first pass selects low timestamp, second pass selects low fan-out
- performs at least as well as LTF
- analyze fan-out structure offline, can use runtime prediction

Furthermore, it is claimed that ARES can eliminate *entire* rollback trees [6]. The end result is that nothing is lost in terms of performance by using ARES instead of LTF. In fact, ARES, as compared to LTF, has a higher percentage of rollbacks (though the rollbacks are shorter), reduces antimessage traffic up to 20%, and has total performance increase of up to 10% [6].

Similarly, conservative simulations are overly pessimistic and have small lookahead, resulting in potential loss of parallelism and lower performance and that time warp simulations are too optimistic and end up rolling-back many events [34]. Causal order (CO) was incorporated into time warp using the "happen before" relationship among events. It is assumed that no optimizations are made [34].

Specifically, the following statement is made with regard to the consistency between causal order and time-stamp order: "CO is said to be consistent with TSO iff when a simulation terminates, for all i, $1 \leq i \leq N$, the CO of the external messages processed on EPA_i is coherent with the TSO of the events scheduled by those messages. That is, for any two events e_{i,t_1} and e_{i,t_2} on EPA_i. scheduled by external messages m_1 and m_2 respectively, if $m_1 \rightarrow m_2$, it must hold that $t_1 \leq t_2$" [34]. Overall, causal order-based time warp (COBTW) reduces the number of rollbacks, with more pronounced effects as message population increases and as the execution time grows.

2.2.5 Protocol and Bandwidth Optimization.

2.2.5.1 Bandwidth Requirements. In examining bandwidth requirements for an En Route Mission Planning and Rehearsal (EMPR), it was found that even for a small number of entities—they were considering three vehicles per aircraft on six aircraft—the bandwidth requirements can be significant. The particular scenario being tested was part of the U.S. Army's Future Combat System (FCS), which requires that vehicles be able to communicate to perform mission rehearsals while in transit via airlift. Peak bandwidth demand with the experimental configuration was between 2.5 Mbps and 4 Mbps and a 200 Kbps wireless channel was necessary for communications between the aircraft [17].

One point which was not considered was scalability. With six aircraft, there are 15 connections required for a fully meshed network. Adding even just a few more aircraft greatly increases the number of required links and, consequently, the amount of total bandwidth. This is not meant to imply that all simulations involve communicating across wireless links from one aircraft to another. However, it is quite clear that scaling even an apparently manageable small scenario can have serious implications for bandwidth requirements.

The bandwidth required for a simulated network running OTB was examined, providing detailed results of the distribution and behavior of the PDUs [30]. Most importantly, it was found that with bandwidth above 200 Kbps, the travel time approaches optimal and the peak message queue length is 60 for plane 0 and 40 for the satellite ground station [30]. This means that even with sufficient bandwidth to accommodate all the PDUs that need to be sent some are still queued, presenting a delay in their routing.

2.2.5.2 Interest Management and Filtering. One possible solution to the bandwidth problem is to develop new or improved ways of filtering unwanted, duplicate or erroneous traffic. However, one of the limitations of existing filtering mechanisms is the inability to scale when entities become very concentrated in a small are of the simulated environment [11]. The idea behind approach to resolve this is that entities that group together will have similar interests and can update each other at a detailed level, but entities outside the group do not have the same interests and can be updated about the group as a whole by an elected representative at infrequent intervals [11]. This is essentially the idea that entities grouped closer together can communicate directly, but communicate via proxy to distant clusters of entities.

Entities become members of groups as their areas of interest overlap. If the areas do not overlap, then a group is not formed. If a group has no members, it is destroyed. An entity can only be a member of one group at a time (this helps avoid duplicate messages). There is also a distinction between high fidelity (interaction messages) and low fidelity (position update messages). Election of the representative user is based on the priority. The scheme reduces the number of transmitted messages by up to 18% over the existing proximity-based scheme [11].

There is a way to extend the existing publish/subscribe mechanism within HLA to reduce network bandwidth and CPU utilization, which is novel in that it relies purely on the local federate's ability to control it's own local publish/subscribe mech-

anism [20]. This makes it possible to implement without depending on coordination with other federate developers. This is accomplished by the subscribing federate distributing the filter rules across the RTI to the publisher [20]. The architecture allows arbitrary filtering, isolates publishers from subscribers and allows reuse of filters [20].

However, it appears that the publisher must call the filtering logic, which then notifies the RTI if it decides that the subscriber wants to see it [20]. This seems to imply that the filter supplants the standard subscribe mechanism. It also seems that if a particular publisher accumulates many filters from many different subscribers, then it must call each individual filter and potentially consume additional local CPU resources. However, the consumption of local CPU resources is considered, but no quantification is given of the amount of CPU consumption [20]. They also mention that this is still beneficial in terms of overall processor load and network bandwidth utilization. This approach essentially requires a modification to HLA and the RTI, but it is claimed to be within the spirit of the HLA designers' intent [20]. Unfortunately, there are no experimental results or concrete data, nor the impression that such data are forthcoming [20].

2.2.5.3 Packet Bundling. Though the concept is applicable to network traffic in general, the viability of bundling DIS protocol data units (PDUs) was examined specifically [29]. Since modern networks have a great deal more bandwidth than in the past, a generic bundling approach would likely not be very fruitful. However, given the limited number of types of DIS PDUs, and their fixed format and size, it is much more likely that a bundling approach specific to those few types of packets would be worthwhile.

Specifically, consider a network composed of nodes on an aircraft connected via Ethernet, with the aircraft connected via wireless links to each other and to satellites; the satellites are, in turn, connected to ground stations [29]. Traffic was collected from an OTB simulation and used in the scenario to provide a more realistic traffic pattern than could be obtained via statistical generation. The bundling strategy can

be stated as if matching PDUs arrive within the available time window, they are bundled and sent together [29].

One approach employed a neural network to predict the type of the next incoming PDU to see if it could potentially be bundled and whether or not to wait for it. Analysis of the slack time showed that for the scenario studied, at least 256 Kbps was needed for the wireless links to be able to accommodate the traffic [29]. The best performer reduced the number of transmitted PDUs by 35% and the number of transmitted bytes by 21%.

2.2.5.4 New Protocols. An alternative approach to optimizing existing protocols is the creation of new protocols. One such communication protocol for large-scale virtual environments is called SCORE [18]. The protocol performs transport layering based on cells that identify groups of multicast recipients that are dynamically partitioned to keep the traffic level below a specified threshold. Multicast groups are limited in that there are a limited number of available multicast addresses in the IPv4 scheme and the probability of collision increases as their use spreads. There is also a cost to join, leave and maintain membership of multicast groups. Estimation of the best cell size can either be performed once and remain static, or it can be reevaluated throughout [18].

Since the SCORE protocol was designed from the ground up with the benefit of lessons learned over many years using existing protocols and encountering many other problems and shortcomings. This "fresh" approach allows for a clean implementation that is flexible and able to deal more effectively with difficulties encountered in large-scale virtual environments. The primary drawback to such an approach is the immense installed base of systems using existing protocols. Developers and organizations have mostly already chosen to use DIS and/or HLA. The prospect of adopting a new, unfamiliar, protocol to use in new development efforts is relatively low. The likelihood of those same organizations going back and redeveloping existing systems using a new protocol is even lower. As with any new technology that is a significant departure

from the norm, it will not see widespread adoption until the pain of staying with the status quo is greater than the pain of change.

New protocols need not completely replace existing protocols. They can just as easily be built on top of already existing protocols. This has the advantage that modifications to existing systems can be performed by adding one or more components. However, there is also the potential disadvantage that after enough new features have been layered on, the system begins to acquire a sense of being a patchwork because of the many differing modifications.

There are two kinds of bandwidth reduction [28]:

- data aware: "intelligent marshaling techniques that direct information to interested parties."
- data independent: compression, routing, etc

Data independent bandwidth reduction does not scale and instead effort is concentrated on data aware techniques: simple approach, dead reckoning, network subscription, group subscription, and relevance filtering are considered [28].

simple approach handled by the sender; send individual messages to each interested party

dead reckoning intermittent updates; each process fills in the gaps on its own

network subscription listen to channels where information is broadcast or multicast to those who are listening

groups subscription join a group to get the messages; extended by MPI

relevance filtering similar to the HLA publish/subscribe approach [28]

The concept of "thin agents," which are like traditional agents, but with lower overhead and only called when needed is introduced [28]. Thin agents must be distributed to each node, be provided an execution context and must have access to basic communication services of the infrastructure [28]. Unfortunately, the case study is contrived because updating only every 10 seconds seems unrealistic and they provide no basis for asserting linear message growth with the number of tanks in the simulation. This approach has limited applicability, at best.

2.2.5.5 State Distribution. The problem of distributing state information in logical time simulations has been well explored in the realtime simulation community [25]. Additionally, it is not suitable for federated simulations to use a connection transfer protocol—used to make sure a federate does not receive messages in the past—as is used in other simulations, because federates are permitted to enter and leave the simulation while it is in progress [25]. That is, since a federate can quit the simulation completely or enter part way through, there may not be a place to transfer the connection to or from.

When dealing with moving federates, sender initiated connections are more difficult to handle than other types of synchronized connections. Causality errors resulting from sender initiated connections can be prevented with one of two approaches, each requiring limiting the lookahead of either the sender or receiver [25]. The connection lookahead approach uses a combination of the two previous approaches, where each federate is responsible for specifying its own connection lookahead, limiting how far into the future a connection can be established with another federate [25].

Each cell has a corresponding counter indicating how many entities are subscribed. The counter functions similarly to the hard link counters used in Unix/Linux file systems to decide whether or not a file "exists." When a region's attribute is updated, the federate searches its log. Since each federate maintains a separate log, the search can be accomplished quickly. Evaluation of the synchronized DDM system consisted of entities conducting random walk with a 1-cell update region and a 6x6-cell subscription region. Performance was good, depending on log densities, but no more than 8 federates were used for performance analysis [25].

Another viable area for optimization is the sheer number of synchronizations of state information that must take place. Synchronization is optimized where the simulation may experience large differences in event density (e.g., peace time and war time) without adversely affecting the simulation [27]. It was found that at higher event densities, the traditional time-stepping approach performs better and that adapting

between the two, depending on simulation conditions, is recommended as a viable alternative [27].

They also introduce a concept called "super-stepping," which works by only notifying the process about a time step when there is an event waiting for that processor to process it. This method uses barrier synchronization. In all cases, super-step boundaries, overhead events and time advances, the piggy-backed super-step approach provided significant improvement over the conventional time-stepped approach. The super-step value must be bounded above by a threshold to prevent its value from getting too large and adversely affecting performance because of inability to respond to changes in event density [27]. The improvements were an approximate reduction of 60% to 20% of the original traffic.

Further, there is a low overhead technique for computing the utility of an entity and determining the update frequency of its state information to other entities based on the measure of utility [35]. Other methods (relevance filtering and dead reckoning) may not be enough when the simulation becomes very large with tens of thousands of entities [35]. Utility is partly determined by influence (how many entities can be influenced by a particular entity) and proximity (how close other entities are to a particular entity). The utility is based on the relative utility of the entity with respect to the other entities of its type and the area of influence (which can be divided into multiple levels depending on the distance from the entity).

Details are provided on how a federate can make use of this technique by subscribing to dummy attributes and periodically refreshing the associated regions to keep the utility value current as the entity moves [35]. The experiment, however, is limited and shows that as the number of levels into which the AOI is divided increases, so does the overhead (at a linear rate); the overhead goes down as the number of entities increases [35]. The overhead is measured by measuring the time it takes to execute the simulation without the utility calculation and the time with the calculation. This implies that the simulation was executing as fast as possible. They

don't provide a detailed or explicit explanation about how the updates slowed down in their experiment.

2.2.6 Other Potential Areas for Improvement. The topic of bandwidth utilization and optimization has been examined [30], [17]. The investigations dealt specifically with the U.S. Army's OneSAF, which uses DIS as one of its protocols. The scenarios which they examined had relatively modest bandwidth requirements, and their conclusions revealed that once there was enough bandwidth, adding more did not help the latency or delay (they were examining systems connected through wireless or satellite networks).

In a different instance, the idea of bundling PDUs being sent between entities which were connected via wireless links was examined [29]. Specifically, the PDU traffic was analyzed and a neural net operating over a sliding window of PDUs was used to identify like PDUs and bundle them together. The bundling approach taken by involves a sort of compression. That is, by grouping similar or identical PDUs, they are able to transmit something which still represents both PDUs but does not require as much space. This approach is necessary given the low capacity links they were investigating. The bundled PDUs were then unbundled on the remote end.

2.2.7 Ethernet Frames. Ethernet frames, also called packets, have a variable size. Other physical networking technologies, such as asynchronous transfer mode (ATM), rely on small fixed size cells. The variability in the size of Ethernet packets can be used to advantage in networks carrying DIS simulation traffic. This is because DIS PDUs tend to be small. Much smaller, in fact, than the available payload in an Ethernet frame (this will be discussed in detail in Section 4.2). As seen in Figure 2.1, the header of an Ethernet frame is fixed at 14 bytes and the trailer is fixed at four bytes. The payload, which can carry any higher level protocol and associated headers and metadata, can range from 46 bytes to 1500 bytes.

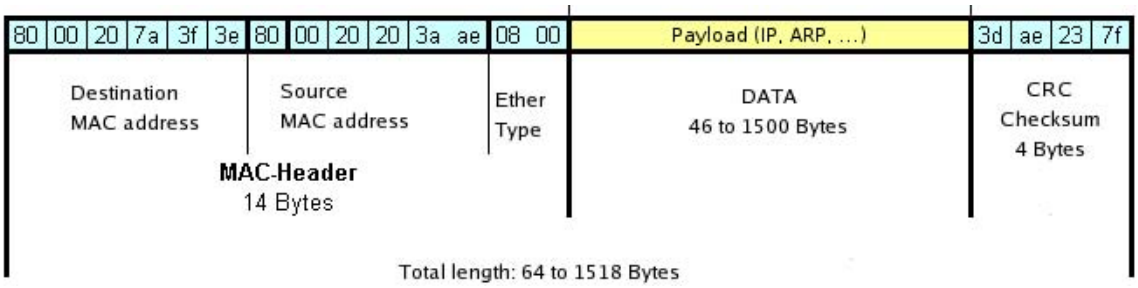

Figure 2.1: Ethernet frame format [31]

2.3 Summary

Distributed simulation has been around long enough now that the foundations have been laid down and the state-of-the-art progresses to more advanced areas of research. Clearly, the state-of-the-art has been built upon greatly from the foundations laid by Fujimoto and other pioneers in the field. Some issues, such as timekeeping and causality, have been explored in depth and are well understood. Other issues, such as scalability and aggressiveness, are not as well understood or not as optimized and the subject of much ongoing study.

For example, managing state distribution is something that is handled well in most small-scale simulations, but it becomes somewhat of a burden as the simulation scales up. As another example, load optimization is an area with a multitude of different, equally effective, approaches. These and other challenges provide ample source for research in the area of distributed simulation. The background presented here is not intended to be comprehensive. However, it is intended to provide a snapshot of where the current state of the art is leading to the work and findings presented in this thesis.

III. Preexisting System

This chapter presents the network and the filters used at the DMOC. That is, the description is one of the operation and design of the filter prior to any modifications proposed in this thesis. The discussion of the physical network topology is presented for the sake of completeness.

3.1 Network Topology

The real network topology used at the DMOC is quite complex. Since this thesis is not concerned with all of the fine details, Figure 3.1 shows a generic representation of the network deployed at the DMOC.

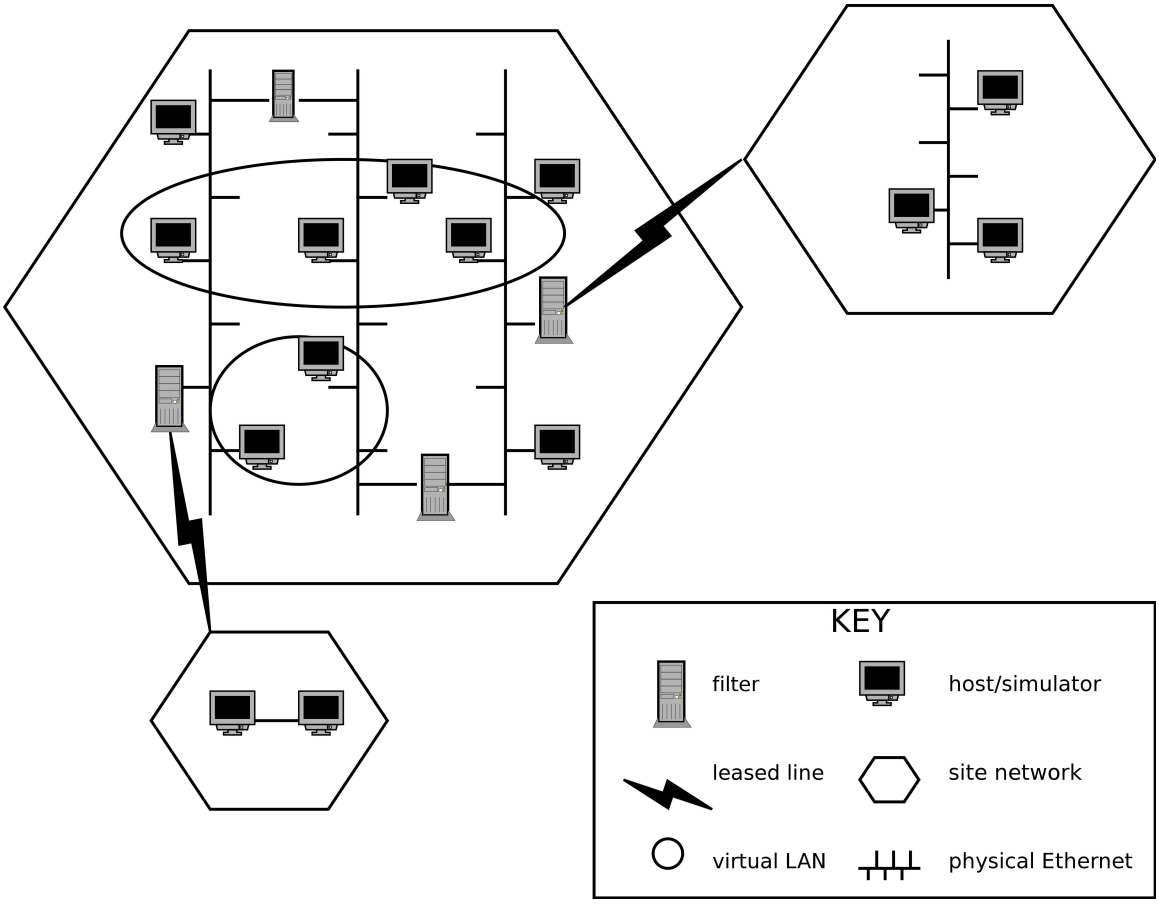

Figure 3.1: Representative model of DMOC VIRTUAL FLAG exercise network

3.1.1 Physical Network. The largest network in Figure 3.1 represents the local network at the DMOC, with the two smaller networks representing remote sites that connect via leased lines. The leased line connections are T1, thus providing 1.544 Mbps symmetric upstream and downstream bandwidth. Additionally, there are other sites that are connected via high-bandwidth networks, such as the Defense Research and Engineering Network (DREN) and Joint Training and Experimentation Network (JTEN).

Each remote site can have one or more simulator systems that are connected to the exercise. Additionally, other exercise-related traffic (e.g., configuration files, voice and video conference) travel over the link. In practice, the filtered simulation traffic accounts for about 800-1200 kbps, leaving sufficient bandwidth for the additional traffic. In the case of sites connected via higher speed networks, this is not a concern [23].

3.1.2 Virtual Networks. Within the DMOC there a number of physical network segments. However, the systems are logically connected via a number of virtual LANs (VLANs). This allows simulators to be "grouped" functionally, only routing specifically requested traffic between the VLANs when necessary. These VLANs include the systems that are connected from remote sites. For example, there is one VLAN in particular, the so-called "Ground LAN," to which all ground entities are connected. If there is an air entity which requires traffic specific to a particular ground entity, then that traffic is brought in from the corresponding VLAN to the Air LAN.

In practice, the VLANs were named for the primary focus of the entities connected to them, e.g., air, ground, UAV, etc. Presently, nearly all entities resident at the DMOC are connected to the Ground LAN and the various VLANs retain their historical names. This approach allows nearly all of the traffic to be broadcast over the Ground LAN, which is convenient for logging. Traffic to the other VLANs can be filtered as necessary, while all traffic is allowed into the Ground LAN. For example,

nearly all remote site connections over leased lines are connected through filters to the Ground LAN.

3.2 Routing and Filtering

The PDU traffic on the network at the DMOC is routed and filtered. That is, the "DIS Filter Software," is configured to route traffic from the network on one interface to the network on another interface. In this respect, it functions like an actual router, though entirely in user space. At the same time as it is routing PDUs, the software is also filtering each individual PDU according to the algorithm and specified rules.

3.2.1 Traffic Composition. In general, PDUs are broadcast across the network at a relatively constant rate. This stems in part from the requirement by the DIS standard that entity state (ES) PDUs be broadcast at a fixed interval by all entities. In the data sets used for the research in this thesis, comprising logged data from previous VIRTUAL FLAG exercise and Multi-Service Distributed Events (MSDE), the two most common types of PDUs were entity state and signal PDUs. Signal PDUs usually accounted for 40% of the number of PDUs in the simulation, while entity state PDUs accounted for between 40% to 50% of the number of PDUs in the simulation.

3.2.2 Filtering Algorithm. There are five components to the base filter algorithm which determines whether or not a PDU is allowed to pass the filter. The checks are:

1. exercise ID
2. PDU type
3. entity ID
4. enumeration
5. range

The exercise ID check is performed to ensure that the PDU being examined belongs to the particular exercise for which the software is routing traffic. The DIS standard allows for multiple exercises to run over the same network, so the PDUs are required to identify the exercise to which they belong. If a PDU belongs to another exercise, there is no need to examine it any further.

The PDU type check allows the filter to choose whether or not to allow the PDU to pass based on whether the operator has allowed PDUs of that type. The DIS filter application allows the operator to choose whether or not to allow each of 16 different types of PDUs.

The entity ID check tests whether the site, application and/or entity fields match any provided by the operator. It is possible for the operator to specify the three fields, using a wild card character for zero or more fields, in order to specifically allow or deny PDUs that match.

The enumeration check is rather simplistic. The operator is able to specify a list of zero or more entity types and/or entity IDs which should be allowed to pass the filter. Additionally, the operator can specify to allow all entity types and entity IDs, or likewise to deny them all. If the enumeration filter has not been set to allow all or deny all, then the filter simply uses the built-in iterator object's `find()` function.

The range check allows the operator to specify a cylindrical or spherical shape inside of which traffic will be passed. The shape is denoted by a latitude and longitude to specify the center point, a range in nautical miles to represent the radius and an altitude in feet. When the range filters are enabled, each PDU that passes the above checks will then be examined to see if it is inside at least one of the specified range filters. This decision is made by using the location of the PDU (given in latitude and longitude within the PDU itself) to determine the distance. If the distance from the PDU to the center of the specified sphere or cylinder is within the radius, then the PDU is passed. If not, then the PDU is discarded and processing goes on to the next PDU.

3.2.3 Filter Rules. The algorithm itself requires a set of specified filter rules. Section 3.2.2 alluded to some of these rules in the form of selecting which types of PDUs are allowed and which ranges or enumerations are specified. The selection of the exact filter parameters depends on the particular needs of the exercise and the network. For example, if a site is participating with only a small number of aircraft that are grouped together closely, it would be possible to choose a more restrictive range filter.

In actual usage, the filter rules are planned in advance. The filter settings chosen are based on knowledge of the parameters of the exercise and experience of the engineers with past exercises. Additionally, once the exercise actually starts, the rules can be manually tweaked in response to requests from participants or in order to reduce the amount of unnecessary traffic that is being passed by the filter.

IV. Proposed Changes

4.1 Adaptive Range Filtering

The idea of adapting the range-based filtering functionality came from a discussion with DMOC engineers [23]. More than one interviewed engineer expressed a desire to see the static nature of the range-based filtering "upgraded" to something dynamic. This would allow a greater degree of control over the number of PDUs allowed to pass by the filter. This can be done with a small amount of processing overhead, especially if the adaptivity is kept simple.

4.1.1 Basis for Adaptive Range Filtering. The general idea behind adapting the range-based filter is that certain events may necessitate or allow for a change in the area originally specified by the operator. For example, as the time of day changes and night falls or day breaks, visibility would change, requiring a corresponding increase or decrease in the radius of the range. Another possibility is that the center point of the area should move to follow a particular entity. It is possible to take advantage of the ability to adapt the range filter by specifying a range filter that is suitable at the start of the simulation or exercise, but which can be modified during the event in order to remain suitable throughout. Choosing more conservative ranges that can then be adapted can eliminate the need to specify overly large areas or to manually tweak areas when an entity travels outside of its originally forecast area.

Making a more conservative range choice ultimately leads to a requirement to periodically check whether the range needs to be adjusted. By comparing a PDU currently being processed with the previous state of the range, the filter can decide whether the range needs to be adjusted at any point in time. In the case of a range being adjusted for changes in the time of day, the timestamp is recorded and when a future PDU is checked against the range filter, its timestamp is checked as well. If the difference in the timestamps reveals that it is necessary to adjust the radius of the filter, then the radius is adjusted as necessary. Only if the radius is adjusted is the new timestamp recorded and used for the next comparison. This prevents network

jitter, which can be a problem with UDP packets arriving out of order, from causing multiple adaptations in a very short period of time if PDUs arrive out of timestamp order.

The nature of providing an adaptive range-based filter mechanism within the already existing framework of the static filter mechanism makes it possible to further reduce the amount of PDU traffic forwarded by the filter. Various different approaches are possible, including adjusting the range based on proximity of other specific entities or entity types, adjusting the position of the range in response to some event, or even something else. Each type of adaptive behavior requires a different amount of processing and overhead in the form of maintained state.

The processing and state overhead required to implement a particular algorithm must be carefully evaluated in the context of the specific simulation event. This is especially true since the data being used has already been logged and much of the information required to evaluate the suitability of a complex adaptive algorithm is not available. It is beyond the scope of the work presented in this thesis to conduct such an analysis. The algorithm described below is neutral to the particular form of adaptivity used. However, it is described along with the particular adaptivity presented in this thesis.

4.1.2 Modified Algorithm. Figure 4.1 provides the pseudocode for the adaptive range-based filtering process.

```
if (adapt_range_filter_enabled)
    check PDU header for timestamp
    if (timestamp is absolute)
        if (hour has lapsed)
            adjust radius in each affected route
            record timestamp of current PDU as new timestamp
```

Figure 4.1: Adaptive range-based filter algorithm

The first step is to check if the operator has enabled adaptive range-based filtering. This algorithm does not modify the range checking itself, rather it is an

augmentation which simply modifies the specified range as necessary. This is made possible because the same function, which is initially used to set the location of the range and configure the associated filter parameters, can be called again from elsewhere in the code. Additionally, the radius itself is kept in a variable in memory throughout the execution of the program and is easily modified.

The adjustment of the radius is the specific implementation that was tested in this thesis. Tests were conducted using a simple constant reduction of the radius by 20 nautical miles, every hour on the hour. This can be further modified to compute factors like the amount of visibility lost at particular times of the day or night. Likewise, weather could be used as a factor in determining the amount of adjustment, which requires additional overhead.

4.1.3 Expected Benefit. With respect to range-based filtering itself, the effectiveness is dependent on the initial choices of the operator and the simulated area being encompassed. This is even before considering any adaptations. Thus, it is possible to choose an area sufficiently large that any future adaptations fail to eliminate any additional traffic. It is also possible that the initial range is chosen in such a way that after one or more adaptations, much "relevant" traffic is being excluded. It is also possible that a range that is chosen based on the location of two entities becomes invalid if the two entities move apart. It is not within the scope of this work to tackle these issues, however they do bear mentioning.

It is expected that with a good choice of initial center point and range and suitable adaptivity, this approach will provide measurable benefit. The benefit would be realized in the form of reduced PDU traffic passed by the filter with little or no negative impact to the participants receiving the filtered traffic. The reduction in PDU traffic can then be translated directly into a reduction in the number of bytes sent across the network. As previously stated in Section 1.6, the static filtering approach implemented at the DMOC already provides a significant reduction in PDU traffic.

This approach seeks to further improve the performance with an added flexibility that can make use of different adaptivity approaches as the situation requires.

4.1.4 Suggested Application. Given the need to develop an additional algorithm to plug into the adaptivity framework, significant analysis and development would be required on the part of the engineers overseeing the simulation. While this cost exists in order to first make use of the adaptivity, it is certainly possible to reuse prior implementations that have been proved to work. Additionally, it is also possible to build a number of different algorithms into the system and choose from among them via a runtime configuration option.

The modified range-based filtering can also be applied to the previously existing filter without any modification to any simulator system or the network infrastructure. As one of the goals of this work was to provide simple and minimally intrusive improvements, this clearly fits the criteria.

4.2 PDU Bundling

Examining the DIS standard shows that many PDU types are very small in size [12]. For example, entity state PDUs are 1280 bits (160 bytes) and signal PDUs are 272 bits (34 bytes). Contrast this with the 1500 byte maximum transmission unit (MTU) of a standard Ethernet network, and it is clear that sending individual PDUs does not make efficient use of the available payload in an Ethernet frame. Further, when comparing the amount of payload used to the amount available, only a small fraction is used. It is clear from this simple inspection that there is an opportunity to greatly reduce the amount of overhead by grouping PDUs together into a single Ethernet packet.

4.2.1 Basis for PDU Bundling. The link capacities examined by Vargas, et. al., were 64 kbps, 256 kbps, 512 kbps and 1 Mbps [30]. The lower capacity links, which comprise the bulk of their investigation and analysis, are not particularly relevant to

the "worst case" situation being examined in this thesis, which is a dedicated T1 line connecting to a distant remote site. Regardless, the idea that PDUs can be bundled together for transmission over a lower capacity link and later unbundled on the remote end is an excellent one.

While the neural network and compression approach was appropriate for Vargas, et. al., it would introduce too much additional complexity to the filter software. The primary reason for the lack of suitability is that the modifications proposed in this thesis are meant for an existing application-level packet filter. Attempting to graft a neural net and complex compression scheme onto such a system would likely required some significant changes which would likely not be acceptable by the developers at the DMOC. Additionally, as mentioned in Section 3.1.1, the filtered traffic typically stays in the 800-1200 kbps range, which provides a suitable buffer.

Even with the unsuitability of the neural net and compression approach, there is one idea which is very applicable to the situation being studied here. Specifically, that of bundling, but without compression. By creating a queue and accumulating PDUs until some predetermined moment, it is possible to create network packets which take greater advantage of the available payload capacity of an Ethernet and also greatly reduce the overhead wasted from sending many small packets. This has the effect of optimizing the payload utilization and reducing the total amount of data transferred.

4.2.2 Bundling Algorithm. There are two components that are important to the bundling algorithm. First, the Ethernet frames and their construction will help to show how this approach is suitable. Second, the implementation of the bundling algorithm will show how this approach can realize a benefit.

Since DIS PDUs are sent using the user datagram protocol (UDP), the associated headers must also be carried as part of the payload. The header of an IPv4 packet is 20 bytes and the header of a UDP packet, which rides over IP, is eight bytes. Coupled with the 18 bytes of Ethernet packet overhead, the amount of overhead becomes 46 bytes for each UDP packet sent. The DIS protocol headers are not counted

for this computation since they are unavoidable for each PDU, and they are used at the application level.

As shown in Table 4.1, entity state and signal PDUs make poor use of the available payload. In a DIS simulation, entity state PDUs are transmitted periodically to broadcast to all other entities the location and condition of the sender. The condition can include speed, heading, altitude, and other relevant details. Signal PDUs are used to send information about radio, voice or data signals. The "max" PDU listed in the table is not an actual PDU type, but rather meant to illustrate the utilization possible by making maximum use of the available Ethernet payload. For a 160 byte entity state PDU, 46 bytes of overhead are required. That represents an overhead of 22%, with 78% of the transmitted data being useful at the higher level application layer. The situation for signal PDUs is markedly worse. A 34 byte PDU sent with 46 bytes of overhead represents merely 42.5% of the transferred data being useful.

Table 4.1: Overhead and utilization of Ethernet frames by PDUs (sizes in bytes)

PDU	**size**	**overhead**	**% overhead**	**% useful data**	**% payload utilization**
ES	160	46	$\frac{46}{206} = 22\%$	$\frac{160}{206} = 78\%$	$\frac{160}{1472} = 11\%$
signal	34	46	$\frac{46}{80} = 57.5\%$	$\frac{34}{80} = 42.5\%$	$\frac{34}{1472} = 2.3\%$
max	1472	46	$\frac{46}{1518} = 3\%$	$\frac{1472}{1518} = 97\%$	$\frac{1472}{1472} = 100\%$

If the Ethernet frame were completely filled to capacity with a 1472 byte payload (as shown in the last line of Table 4.1), leaving 28 bytes for the UDP and IP headers and 18 bytes for the Ethernet overhead, then 97% of transferred data is useful to the higher level application. This represents the best possible utilization of an Ethernet frame for sending over UDP/IP. The other PDU types fall in the range between the entity state and signal PDUs. While it would be very challenging to achieve such a high level of utilization in the real world, it is certainly possible to improve on the situation over sending a single PDU at a time.

4.2.3 Queue Wait Time. Having already discussed the issue of increasing the payload utilization in the Ethernet frames, the next issue is that of latency. Any implementation of the above idea must deal with latency in an acceptable way. That is, DIS simulations are sensitive to latency, and the DMOC's filter software has been optimized to introduce minimal latency. Anything that introduces a large amount of latency would likely not be operationally feasible. To that end, the implementation described in this section is based on a maximum "wait time" of 10 ms.

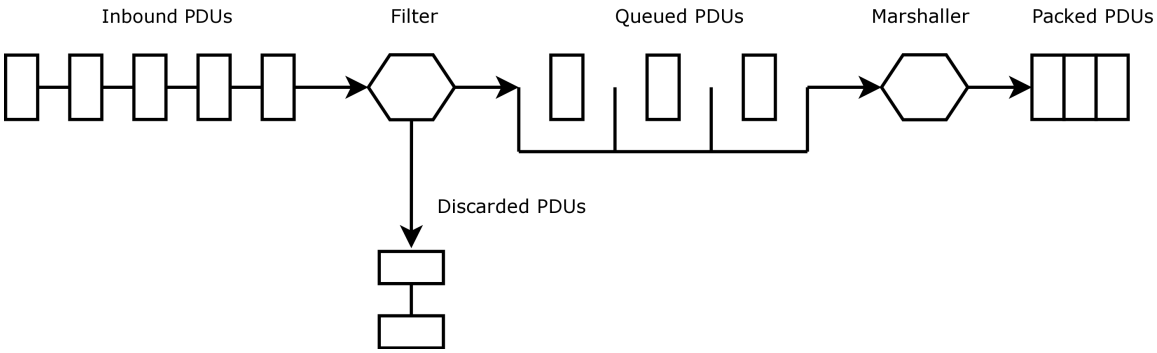

Figure 4.2: Filtering, queuing and bundling process diagram.

Figure 4.2 illustrates the functioning of the proposed system. PDUs arrive at the filter from the local network. The filter examines each PDU and then decides whether to forward or to discard each PDU. The decision to forward or discard is based on the rules specified in the filter configuration, which are dependent on the particular exercise and filter instance. Once the filter decides to pass a particular PDU, the PDU is placed into a queue. This is a departure from the original operation of the filter, where once a PDU was forwarded by the filter it was immediately placed on the network bound for the remote end of the link. The PDUs wait in the queue and every 10 ms the marshaller awakens and begins its process. The marshaller dequeues the PDUs one at a time until a either the queue is empty or the packet payload is full and then sends the packet.

The following brief explanation shows why the 10 ms interval for the marshaller is a good starting point. Suppose that, as stated in Section 3.1.1, bandwidth utilization leaving the filter is approximately an average of 1000 kbps. That is, the queue

is receiving input from the filter and being serviced by the "bundler," as shown in Figure 4.2. Further suppose that the average size of a PDU is about 100 bytes. Since, as stated in the previous section, signal PDUs are 34 bytes and entity state PDUs are 160 bytes for an approximate average of 100 bytes, with entity state and signal PDUs each making up about 40% of the total PDU traffic. Assuming a constant and evenly distributed flow of PDUs, this makes the average departure rate of PDUs approximately 1280 PDUs per second, or approximately one PDU every 0.78 ms. Since 14 PDUs of average size would fill the available payload, the wait time would need to be approximately 11 ms. By waiting for approximately 11ms, it is possible to accumulate enough PDUs to nearly fill the available 1500 byte payload without causing PDUs to backup beyond that.

Thus the 10 millisecond maximum "wait time" initially mentioned was chosen since it provides a balance between trying to fill the available payload and introducing only a minimal amount of latency. For example, to send the smallest PDU, a signal PDU, takes 34 bytes of payload and 46 bytes of overhead, for a total of 80 bytes or 640 bits. A T1 line has 1536 kbps of available bandwidth. That means that transmitting takes approximately 0.42 milliseconds. Assuming no waiting time, transmitting 10 such packets would take 4.2 milliseconds.

Now consider sending the same 10 PDUs in a single packet. This packet would have a size of 406 bytes or 3248 bits (340 bytes of PDUs; 20 bytes of separators between PDUs; 46 bytes of Ethernet overhead). Transmitting that packet would take only 2.1 milliseconds. However, in this second case there is waiting time of up to 10 milliseconds from the time the first PDU arrives until the send operation commences. There is an additional propagation delay based on the distance the information must travel, however this is negligible.

Thus, while the receiver on the remote end must wait 4.2 milliseconds to receive all 10 signal PDUs that are sent separately, sending them together requires 12.1 milliseconds. However, this only considers the delay in transmitting the data once

all other processing has occurred. For example, this does not consider the delay encountered during the first transmission to the filter, nor the delay in filtering the PDUs at within the filter and the delay in processing the PDUs on the remote end. Once these delays are accounted for, added cost of approximately 8 milliseconds is not as significant.

4.2.3.1 Implementation. The actual implementation of the bundling algorithm is relatively straightforward and is shown in Figure 4.3. Essentially, there are three parts to the implementation. The first part, which is not shown, sets a timer to go off every 10 ms and raise a SIGALRM, then makes use of the the `signal()` system call to set a signal handler that marshalls the PDUs out of the queue, bundles them and sends them.

The `Marshall_And_Send` algorithm is relatively straightforward as well. Essentially, as long as there are PDUs left in the queue, it will dequeue them one at a time, and try and pack them into the send buffer until the buffer fills. Once the next PDU will overflow the buffer, the buffer is sent and cleared and the process starts over. Since the signal is raised by the operating system based on the counter setting it off every 10 milliseconds, this process happens regularly. Additionally, in order to prevent unnecessary back ups in the queue, once the flush operation starts, the queue is flushed as many times as needed until all PDUs have been sent. Thus the longest that any one PDU will wait in this stage is 10 milliseconds plus the overhead of actually delivering the packet to the operating system's networking code.

4.2.4 Expected Benefit. While bundling will not by itself reduce the number or size of the PDUs transmitted, it does help to make more efficient use of the existing network transport. This means maximizing use of network packet payload while still minimizing the amount of delay introduced to accomplish this.

4.2.5 Suggested Application. While the proposed bundling approach has the potential to greatly reduce the amount of overhead and total traffic, it does come at a

cost. The DIS standard does not provide for a way to send multiple PDUs together. This leaves two possibilities for being able to decode the bundled PDUs. First, each simulator could be modified to recognize the packed together PDUs and unmarshall them. This is unlikely because, as previously stated, the number of deployed legacy systems makes any universal modification nearly impossible. The second option is to install an additional filter on the remote end of the link. The purpose of that instance of the filter would be to unmarshall the bundled PDUs and broadcast them on the remote network.

The second approach is much cleaner because it allows this implementation to remain completely transparent to the simulators participating in the simulation. Furthermore, the deployment of additional filter machines on the remote networks opens the possibility for performing the same bundling on traffic which originates on the remote network. Without an instance of the filter running physically on the remote network, it is not possible to likewise bundle the traffic originating there without modifying all simulators to be able to handle that format. Developers generally prefer to keep to established standards, so the second approach is the recommended course to implement this bundling method.

```
Send_PDU:

if (bundle_pdus_enabled)
    place PDUs to be sent into queue

Marshall_And_Send:

clear send_buffer
clear pdu_count
while (queue_not_empty)
    dequeue
    if (PDU would overflow send_buffer)
        send send_buffer over network
        log number of PDUs and bytes sent
        clear send_buffer
        clear pdu_count
        if (PDU is too large to fit in send_buffer)
            send PDU directly
            log 1 PDU and bytes sent
        else
            place PDU into send_buffer
            place two byte separator into send buffer
            increment pdu_count
    else
        place PDU into send_buffer
        place two byte separator into send_buffer
if (send_buffer not empty)
    send send_buffer over network
    log number of PDUs and bytes sent
    clear send_buffer
    clear pdu_count
```

Figure 4.3: PDU bundling algorithm

V. Experimental Data and Analysis

5.1 Description of Data Sets

The data sets used for analysis of the methods proposed in this thesis were composed of logged data obtained from real simulation events. Table 5.1 gives a summary of the sizes and durations of each set. In general, each data set represents a log starting from 10 to 30 minutes (depending on the specific circumstances) prior to the official start of the simulation. The log likewise continues until 10 to 30 minutes after the official termination of the simulation. Each event consisted of as many as 30,000 entities, including live, virtual and constructive.

Table 5.1: Summary of data sets used for analysis

data set	**start time**	**duration**	**total PDUs**
1	13:21:17	2:11:25	1,461,931
2	11:51:57	1:29:26	964,343
3	11:35:14	2:29:02	1,749,411
4	11:32:23	3:20:04	2,723,426
5	11:37:27	2:53:30	1,375,946

5.1.1 Impact on Analysis of Scalability. The scale of the simulation events is one of the elements which lends weight to the validity of the results obtained. Many academic publications in the areas of computer simulation and distributed simulation use simple configurations with a few dozen or at most a few hundred entities. While this may serve to show whether or not a particular approach works, it typically fails to address one of the most important elements in distributed simulations: scalability. In the work presented in this thesis, real world data from large distributed simulation events were used in order that the question of scalability might be considered as one of the most important.

5.1.2 Logging of Troubleshooting PDUs in Data Set. The behavior of starting the logging of PDUs prior to the official start and continuing it beyond the official termination is for troubleshooting purposes and is unavoidable in the analysis as there is no way to determine which PDUs are transmitted before, during or after the official

simulation time. Regardless, this is simply manifested as a noticeable increase in PDU traffic not long after the start of the data set and a noticeable decrease shortly before the end. In general, the data sets are themselves composed of approximately 40% signal PDUs and approximately 40% entity state PDUs. The remaining approximately 20% is composed of smaller numbers of fire, detonation, IFF, emit, transmitter and receiver PDUs.

5.2 Description of Experiments and Data Collection

Experiments were conducted using a test network of six workstations running various filter configurations. The available logged PDU data were played back in realtime on the network and each filter performed its function and logged the relevant data to disk.

5.2.1 Experimental Network. The network consisted of six workstations. One workstation ran Windows XP and the RedSim PDU Logger tool. The logger tool was used to playback the logged PDU data. The other five workstations ran Red Hat Linux version 9. Of the five Red Hat workstations, one ran the updated filter performing only PDU bundling (i.e., without any range-based filtering). The second Red Hat workstation ran the updated filter performing only adaptive range-based filtering (i.e., no PDU bundling). The remaining three Red Hat workstations ran the updated filter and performed PDU bundling *and* adaptive range-based filtering. These descriptions of the machines correspond to the descriptions of machines *a*, *b* and *c*, respectively, given in Section 5.4.

5.2.2 Experimental Procedure. The procedure for each test run consisted of the following steps:

1. Examine the group of logged PDUs to determine the general latitude and longitude locations of the exercise activity.

2. Configure a suitable sphere for the each machine doing the adaptive range-based filtering. (A suitable sphere was centered near the majority of the activity with a range of 60 nautical miles to start.)
3. Start the filters.
4. Load the logged PDUs into the logger tool and begin playback.
5. When playback is complete, stop the filters and recover the logged data.

The data that were logged to disk consisted of timestamps (recorded after every 1000th packet sent), number of PDUs sent (one if the PDU was sent individually or more if there was a bundle) and the total number of bytes sent in the packet. The total number of bytes recorded was only those bytes at the application layer and does not include the 46 byte overhead for UDP/IP over Ethernet. The data were post-processed using a custom developed Python script which determined such metrics as total PDUs sent/received, total bytes sent/received (with and without overhead) and amount of overhead saved by the bundling process.

5.3 Adaptive Range Filtering

The idea of using some kind of adaptive algorithm for adjusting the range-based filtration was based on the premise that, by reducing the area of the simulation space that was considered "relevant" to the entities on the remote end of the link, it would be possible to reduce the number of PDUs sent. Possible drivers for the algorithm include changes in the amount of ambient light, the time of day, or any other factor which can be determined by examining PDUs. While there are a large number of possible ways in which this could be implemented, those are not the subject of the work presented in this thesis.

The particular data set used to illustrate the results in this section lasts for 3:20:04. This is the longest data set available and allows for more than one adjustment of the filter during the time of the simulation. This is necessary because of the changes in activity levels at the official start and end of the event, which results in noticeable

changes at approximately those times. By using the longest data set for this analysis, it is possible to ignore the changes caused by the start and stop of the event and focus on the changes effected by adapting the range filter.

5.3.1 Effect on PDU Traffic. Examining Figure 5.1, which shows the effects of only the adaptive range-based filtering on the overall PDU traffic, four large drops in traffic are noticeable. The first, near t = 2,000 (approximately 33 minutes), can be attributed to a transient drop in simulation activity, especially since the traffic immediately begins to rise even more quickly than before. The second drop occurs near t = 3,500 (approximately 58 minutes), the third near t = 6,500 (approximately 108 minutes) and the fourth near t = 8,200 (approximately 137 minutes). The second, third and fourth drops in the rate of PDUs transmitted are followed by periods of level performance, indicating that the changes are likely not caused by transient behavior.

As explained in the previous section, the PDUs are logged for a period of time before and after the actual simulation begins. The last drop in the PDU transmission rate, near t = 8,200, can almost certainly be attributed to the end of the simulation itself. At this time, some simulators would certainly have been shut down, precipitating a drop in the network traffic.

The second and third drops in transmission rate occur during the time period where the bulk of the simulation activity was taking place and are nearly one hour apart. This means that the effects are those of adaptive adjustments to the range-based filter. While the effect is noticeable, it is clearly not dramatic. It should be noted that the original implementation of the filter software from DMOC has been in use for a number of years and has been under constant improvement. As a result, the DIS filter software has been optimized very well. Given the large size of the simulation which produced this data set, this is only a minor improvement. Thus, smaller simulations would likely not benefit even this much from this approach. However, larger simulations would see a larger margin of improvement.

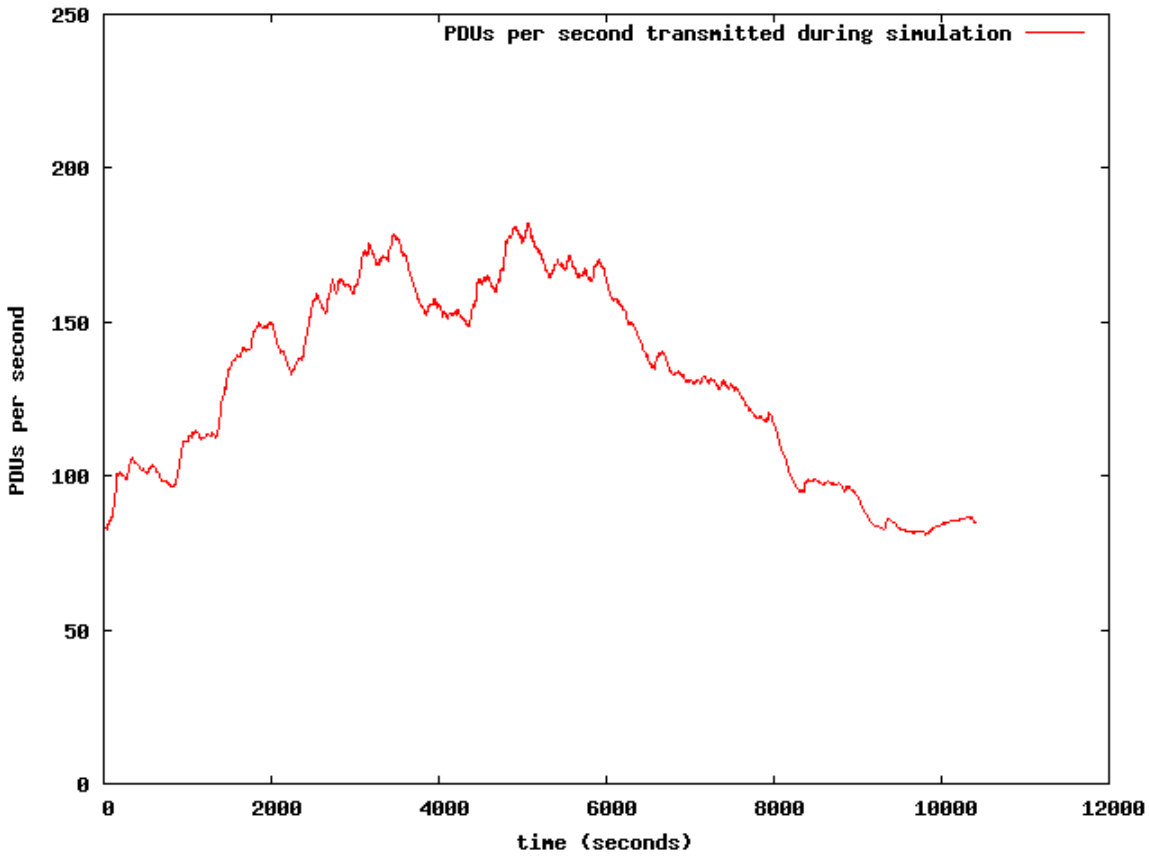

Figure 5.1: PDU transmission rate throughout simulation

5.3.2 Effect on Total Bytes transferred. Beyond the effect on the number of PDUs transmitted, there is also an effect on the number of bytes transferred. Since the PDUs are uniformly distributed, as discussed in Section 4.2.3, the general trend is the same as shown in Figure 5.1. This makes sense as the distribution of PDU sizes is approximately 40% entity state PDUs and 40% signal PDUs, with the rest sufficiently divided among the varying possibilities.

5.4 PDU Queuing

The idea of using a queue to gather a number of PDUs and then send them in groups was based on the idea that by introducing a small amount of delay, we could gather multiple PDUs and then send them more efficiently. First, an examination of Figure 5.2 shows the packets received and the packets sent by each machine for

each data set. The data in the figure represent the number of network packets for each of the first four data sets. The letters (a, b, c) represent the identifier for the machine, while the numbers (1, 2, 3, 4) identify the data set. The asterisk (*) for *a3* indicates that data were not collected from that machine during that run due to a system failure.

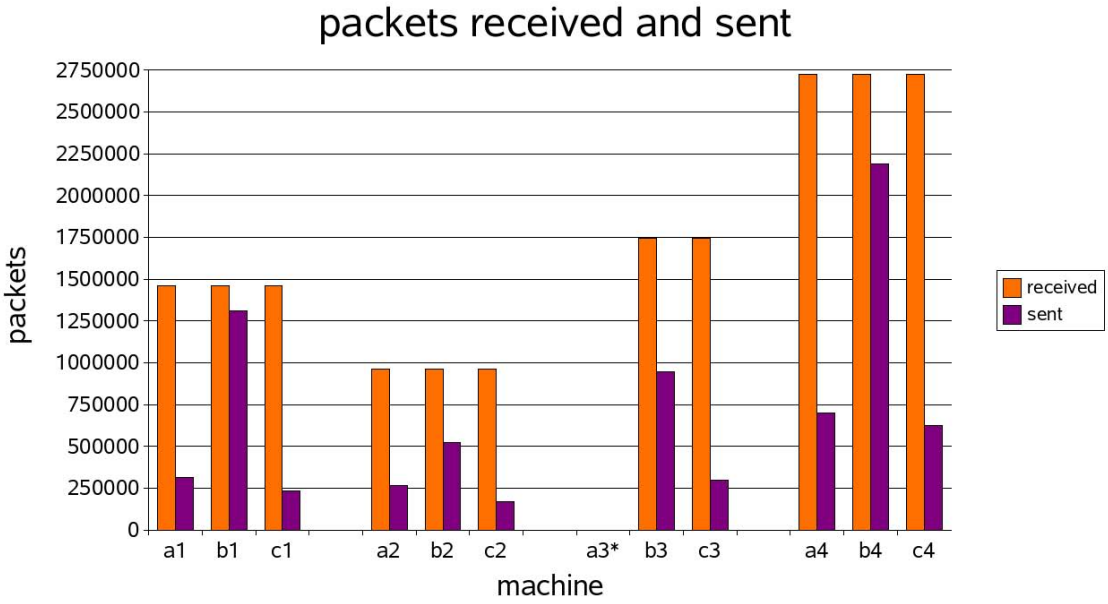

Figure 5.2: Count of network packets received and sent by each machine

The roles of the machines were as follows:

a queuing only (no range-based filtering)
b range-based filtering only (no queuing)
c queuing and range-based filtering

The received packet count is equivalent to the received PDU count, as each PDU arrives at the filter in its own network packet. The sent packet count represents the number of network packets transmitted. For machine *b*, this number represents the number of PDUs sent, as no queuing was taking place. For machines *a* and *c*, this represents the groups of PDUs sent as a single entity in a network packet.

As expected, each machine received nearly exactly the same number of PDUs for the duration of the simulation. Also as expected, machine *b* always sent out fewer

PDUs than it received. This is because machine b was configured to filter out some traffic.

The surprising result was the reduction in the number of packets sent out when the PDUs were bundled. Table 5.2 shows the reductions obtained on machine a, which performed only queuing with no filtering. In each case, there is a dramatic reduction in the amount of overhead expended to transmit the same overall number of PDUs.

Table 5.2: Reduction of packet transmissions for unfiltered PDUs

data set	**received**	**sent**	**reduction**
1	1,457,792	315,135	78.4%
2	961,229	264,104	72.5%
3	no data	no data	
4	2,722,640	701,966	74.2%

An examination of Figure 5.3 shows the reduction translated into bytes of overhead. As established in the previous chapter, each Ethernet packet requires a fixed overhead of 46 bytes in order to send. Thus, the longer bar for each machine represents the amount of *overhead* required to send all the PDUs individually, which translates into 46 times the number of PDUs received. The shorter bar represents the *actual overhead* expended to transmit all of its packets or PDUs. So the overhead expended by machine b in each data set is significantly larger than either that of machine a or machine c, since machine b is not bundling together any PDUs. For machine b, the overhead for packets actually sent is simply 46 times the number of packets sent. On the other hand, machines a and c expended an overhead of 46 times the number of packets sent plus two times the number received. The addition of two times the number of PDUs received is necessary since the packed PDUs are separated by a two-byte separator so the the end of one PDU can be separated from the start of the next.

While the reduction in overhead is dramatic, the analysis would not be complete without considering the effect on the overall traffic on the network. Table 5.3 shows the effect of the reduced overhead on the *total traffic* over the network. That is, it

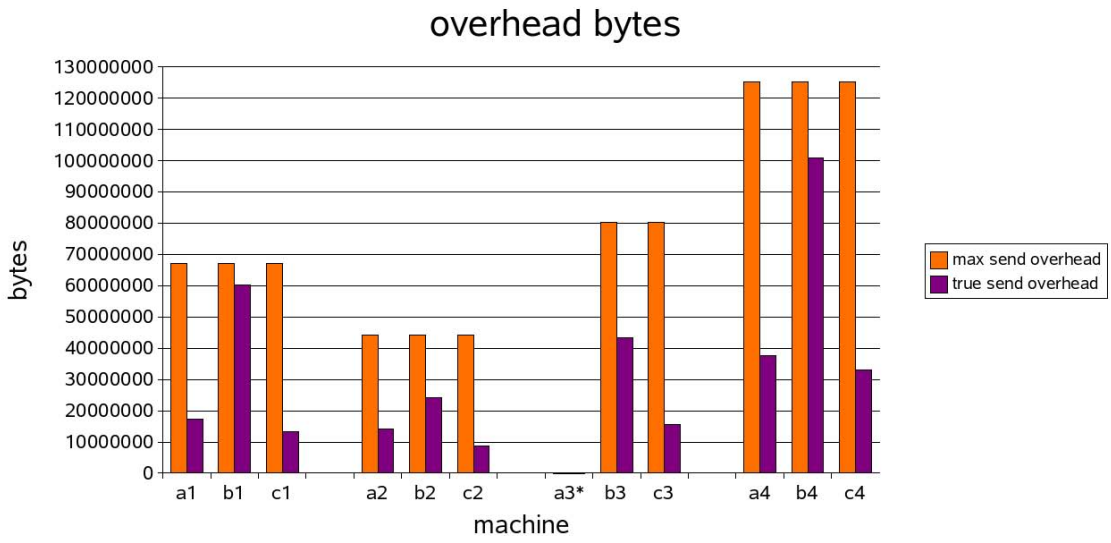

Figure 5.3: Overhead required to send all traffic as received and overhead required for actual data sent

is necessary to examine the effects on the whole rather than the just the part of the overhead. For example, each data set has two lines for each of machines *a* and *c*. Machine *a* performed only queuing of PDUs, while machine *c* performed filtering and queuing.

Thus, for data set one, machine *a* was able to reduce the *total* number of bytes transferred, over the physical layer of the network, by 19.5%. For the same data set, machine *c* was able to achieve a reduction of 20.4%. This is as expected since machine *c* first filtered the PDUs before queuing them. For data set two, the results were slightly reversed with machine *a* achieving a better reduction than machine *c*. This is because machine *c* filtered nearly half of the traffic before before it could be queued. Since the bundling code marshalls and sends every 10 milliseconds, this is an indicator that the packets being sent were not as full as they could have been. The efficiency of the filtering has limited the efficiency of the bundling.

For data set three, there were no results collected for machine *a*. However, the results for machine *c* are in line with expectations based on the performance of the other data sets. Data set four shows performance comparable to data set one, with

Table 5.3: Overhead reduction as a percentage of total bytes transferred

machine	payload bytes	overhead bytes	reduction
a1 (not bundled)	188,827,122	67,058,432	0%
a1 (bundled)	188,827,122	17,411,794	19.5%
c1 (not bundled)	169,224,642	60,262,530	0%
c1 (bundled)	169,224,642	13,415,344	20.4%
a2 (not bundled)	125,648,352	44,216,534	0%
a2 (bundled)	125,648,352	14,071,242	17.7%
c2 (not bundled)	63,768,882	24,045,994	0%
c2 (bundled)	63,768,882	8,736,264	17.4%
a3	no data	no data	
c3 (not bundled)	107,585,082	43,401,138	0%
c3 (bundled)	107,585,082	15,670,354	18.4%
a4 (not bundled)	369,172,068	125,241,440	0%
a4 (bundled)	369,172,068	37,735,716	17.7%
c4 (not bundled)	266,566,412	100,738,574	0%
c4 (bundled)	266,566,412	33,009,556	18.4%

machine *a* achieving a reduction of 17.7% in *total bytes* transferred and machine *c* a reduction of 18.4%.

It should be noted that filtering and bundling *combined* provide the best results. Filtering reduces the total number of bytes transferred by eliminating irrelevant PDUs. Bundling increases the efficiency with which the bytes are transported. Together they achieve a significant reduction in total traffic coupled with an increased efficiency in transport.

5.5 Summary

The data, together with the analysis presented, clearly demonstrate an improvement over the previous version. While each improvement, if taken individually, is relatively minor, taken together they produce a significant reduction in the total amount data transferred and a significant increase in the efficiency with which the data is transferred.

VI. Conclusions

6.1 Applicability of Adaptive Filtering

The adaptive range-based filtering approach is the more expensive of the two proposed changes in that it requires a more complex implementation and more processor cycles. In this respect, it may not be the best choice for the DMOC, given their current utilization of 800 to 1200 kbps of a 1536 kbps link. On the other hand, if utilization were to go higher, the cost could be justified as it would delay the need to take more drastic measures, such as adding additional leased lines at significant cost. Regardless, it would be worthwhile to implement and test in order to ascertain if the benefit could be improved by using more powerful systems to run the filters.

The further requirements of implementing adaptive range-based filtering include creating a suitable algorithm tailored to the specific operational environment of the DMOC. This algorithm would be necessary in order to ensure that the changes to the range filter made sense within the context of a particular exercise and particular external factors. These factors need to be carefully studied with the involvement of the DMOC engineers and with special attention to the projected traffic patterns for the simulated environment being targeted.

6.2 Applicability of PDU Bundling

The bundling of PDUs could be accomplished with a much simpler implementation and fewer processor cycles. However, this imposes the requirement of an additional system being present on the remote end of the link in order to decode the packed PDUs and then rebroadcast them on the remote network. Given the savings demonstrated by bundling PDUs, this appears to be a worthwhile endeavor.

Implementation could be accomplished, and additional systems could be deployed to the remote networks and yet still be administered remotely by the engineers at the DMOC without the need for intervention on the remote end. While there is still a cost for this in terms of additional systems to deploy, this is certainly less expensive than the cost of funding additional dedicated lines to each of the sites.

An additional significant benefit of this approach is that it then becomes possible to bundle traffic originating on the remote network. Currently, it is not possible to affect the traffic prior to its departure from the remote network. By deploying a filter solution on the remote network, it is possible realize the same benefits by bundling traffic bound for the DMOC network. This would likewise make additional link capacity available.

6.3 Overall Conclusions

In the aggregate, the two approaches—adaptive range-based filtering and PDU bundling—presented in this thesis provide a mechanism for improving the network performance during large scale distributed events. Each approach has its own costs and requirements for implementation which make them suitable in some cases and unsuitable in others.

6.4 Future Work

There has been interest in this work from the DMOC and from the Air Force's Simulation and Analysis Facility (SIMAF). The DMOC has committed to incorporating the proposed changes into their baseline for the filter software. This will provide further opportunity for development of these approaches to continue in a production environment. The SIMAF has expressed an interest in finding an application for this work within some of their own products, which are developed to run in distributed environments.

While the work presented in this thesis has shown significant improvements in the targeted areas, there are possibilities for improvement. In particular the following items could be addressed:

- intelligent bundling and compression (as in Vargas [29])
- different max waiting times for queuing PDUs
- changing the range-based filter based on different drivers

The work presented by Vargas included a neural net [29]. While the neural net was deemed as too drastic for the work presented in this thesis, it may be possible to incorporate it, or another form of artificial intelligence, within the framework of the DMOC's filter software. Additionally, the work presented in this thesis considered a maximum queuing waiting time of 10 milliseconds, which showed excellent results. However, examining the effects of different waiting times would indicate whether a shorter or longer delay would have similar, better or worse effects in reducing bandwidth utilization. Further, a different waiting time could provide an indicator as to whether it is possible to dynamically modify the amount of time the marshaller waits before flushing the queue. This would make it possible to wait for shorter periods of time when activity levels are high, in order to reduce the possibility of causing PDUs to wait in a queue. It could also be possible to wait for longer periods of time when activity levels are low, as long as the resulting delay is still acceptable to the simulators on the remote end.

Likewise, adapting the range-based filter can be implemented in a number of different ways. As mentioned in Section 4.1.2, anything from weather, to time of day, to nearby entities can be used to prompt an adaptation in the filter. This requires maintenance of varying degrees of state information. Currently, the DMOC filter software does not maintain extensive state information about the simulation space. However, adding such information could be done within in the existing framework in such a way to make the state information available throughout the filter software. This would facilitate varying approaches to adaptive range-based filtering that depend upon the presence of state information.

Bibliography

1. Boer, Csaba and Alexander Verbraeck. "Distributed Simulation with COTS Simulation Packages". *Proceedings of the 2003 Winter Simulation Conference*, 829–837. New Orleans, Louisiana, United States, 2003.

2. Bowers, Francis A. III and David L. Prochnow. "JTLS-JCATS Federation Support of Emergency Response Training". *Proceedings of the 2003 Winter Simulation Conference*, 1052–1060. New Orleans, Louisiana, United States, 2003.

3. Briggs, Richard A. and Joost Hamers. "Experience in the NATO Pre-Pathfinder DiMuNDS 2000 Federation". *Proceedings of the 1999 Winter Simulation Conference*, volume 2, 1039–1043. Phoenix, Arizona, United States, 1999.

4. Buss, Arnold and Leroy Jackson. "Distributed Simulation Modeling: A Comparison of HLA, CORBA, and RMI". *Proceedings of the 1998 Winter Simulation Conference*, 819–825. Washington, D.C., United States, 1998.

5. Chen, Dan, Stephen John Turner, Boon Ping Gan, and Wintong Cai. "HLA-Based Distributed Simulation Cloning". *Proceedings of the Eighth IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT'04)*, 244–247. Budapest, Hungary, 2004.

6. Cortellessa, Vittorio and Francesco Quaglia. "Aggressiveness/Risk Effects Based Scheduling in Time Warp". *Proceedings of the 2000 Winter Simulation Conference*, 409–417. Orlando, Florida, United States, 2000.

7. Dahmann, Judith S., James O. Calvin, and Richard M. Weatherly. "A Reusable Architecture for Simulations". *Communications of the ACM*, Vol. 42, No. 9:79–84, September 1999.

8. Fujimoto, Richard M. *Parallel and Distributed Simulation Systems.* John Wiley & Sons, Inc., New York, New York, 2000.

9. Gan, Boon Ping, Malcolm Yoke Hean Low, Junhu Wei, Xiaoguang Wang, Stephen John Turner, and Wentong Cai. "Synchronization and Time Management of Shared State in HLA-Based Distributed Simulation". *Proceedings of the 2003 Winter Simulation Conference*, 847–854. New Orleans, Louisiana, United States, 2003.

10. Gottlieb, Eric J., Michael J. McDonald, Fred J. Oppel, J. Brian Rigdon, and Partick G. Xavier. "The Umbra Simulation Framework as Applied to Building HLA Federates". *Proceedings of the 2002 Winter Simulation Conference*, 981–989. San Diego, California, United States, 2002.

11. Han, Seunghyun, Mingyu Lim, and Dongman Lee. "Scalable Interest Management Using Interest Group Based Filtering for Large Networked VIrtual Envi-

ronments". *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, 103–108. Seoul, Korea, 2000.

12. IEEE. "IEEE standard for information technology - protocols for distributed interactive simulations applications. Entity information and interaction". IEEE Standard 1278-1993, May 1993. URL `http://ieeexplore.ieee.org/iel1/2826/6080/00237008.pdf?isnumber=6080&prod=STD&arnumber=237008&arnumber=237008&arSt=&ared=&arAuthor=`.

13. IEEE. "IEEE standard for modeling and simulation (M&S) high level architecture (HLA) - framework and rules". IEEE Standard 1516-2000, September 2000. URL `http://ieeexplore.ieee.org/iel5/7179/19334/00893287.pdf?isnumber=19334&prod=STD&arnumber=893287&arnumber=893287&arSt=i&ared=22&arAuthor=`.

14. Johnson, Glen D. "Networked Simulation with HLA and MODSIM III". *Proceedings of the 1999 Winter Simulation Conference*, volume 2, 1065–1070. Phoenix, Arizona, United States, 1999.

15. Karatza, Helen D. and Ralph C. Hilzer. "Load Sharing in Heterogeneous Distributed Systems". *Proceedings of the 2002 Winter Simulation Conference*, 489–496. San Diego, California, United States, 2002.

16. Kuhl, Frederick, Richard Weatherly, and Judith Dahmann. *Creating Computer Simulation Systems: An Introduction to the High Level Architecture*. Prentice-Hall, Inc., Upper Saddle River, New Jersey, 1999.

17. Leon-Barth, J. C., R. F. DeMara, A. J. Gonzalez, and M. Georgiopoulos. "Bandwidth Optimizations for Integrated Tactical and Training Networks". *Proceedings of the Second Swedish American Workshop on Modeling and Simulation (SAWMAS'04)*, 24–31. Cocoa Beach, Florida, United States, February 2004. URL `http://www.mind.foi.se/SAWMAS/SAWMAS-2004/Papers/P03-SAWMAS-2004-C-Barth.pdf`.

18. Léty, Emmanuel, Thierry, and Françios Baccelli. "SCORE: A Scalable Communication Protocol for Large-Scale Virtual Environments". *IEEE/ACM Transcations on Networking*, Vol. 12, No. 2:247–260, April 2004.

19. Lu, Tainchi, Chungnan Lee, Wenyang Hsia, and Mingtang Lin. "Supporting Large-Scale Distributed Simulation Using HLA". *ACM Transcations on Modeling and Computer Simulation*, Vol. 10, No. 3:286–294, July 2000.

20. Murphy, William S. Jr. and Galen D. Aswegan. "High Level Architecture Remote Data Filtering". *Proceedings of the 1998 Winter Simulation Conference*, 835–840. Washington, D.C., United States, 1998.

21. Nählinder, Staffan. "Similarities in the way we react in a simluator and a real world environmant". *Proceedings of the First*

Swedish American Workshop on Modeling and Simulation (SAW-MAS'02). Orlando, Florida, United States, October 2002. URL `http://www.mind.foi.se/SAWMAS/SAWMAS-2002/Papers/SAWMAS-02-SNahlinder.pdf`.

22. Pham, C. D. and R.L. Bagrodia. "Building Parallel Time-Constrained HLA Federates: A Case Study with the PARSEC Parallel Simulation Language". *Proceedings of the 1998 Winter Simulation Conference*, 1555–1562. Washington, D.C., United States, 1998.

23. Sánchez, Roberto C. Interviews with DMOC Engineers, including Isaac Marquez, Charles Cashulette, Larry Lawrence, Gary Vincent, Mike Mora, Craig Goodyear and Jason Atkinson, July 14-15 2005.

24. Straßburger, Steffen, Thomas Schulze, and James O. Henrikson. "Internet-Based Simulation Using Off-the-shelf Simulation Tools and HLA". *Proceedings of the 1998 Winter Simulation Conference*, 1669–1675. Washington, D.C., United States, 1998.

25. Tacic, Ivan and Richard M. Fujimoto. "Synchronized data distribution management in distributed simulations". *Proceedings of the 12th Workshop on Parallel and Distributed Simulation (PADS'98)*, 108–115. Banff, Alberta, Canada, 1998.

26. Tan, Gary and K.C. Lim. "Load Distribution Services in HLA". *Proceedings of the Eighth IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT'04)*, 133–141. Budapest, Hungary, 2004.

27. Tay, S.C., G.S.H. Tan, and K. Shenoy. "Piggy-Backed Time-Stepped Simulation with 'Super-Stepping"'. *Proceedings of the 2003 Winter Simulation Conference*, 1077–1085. New Orleans, Louisiana, United States, 2003.

28. Taylor, Simon J.E., Jon Saville, and Rajeev Sudra. "Developing Interest Management Techniques in Distributed Interactive Simulation Using Java". *Proceedings of the 1999 Winter Simulation Conference*, volume 1, 518–523. Phoenix, Arizona, United States, 1999.

29. Vargas, J., R. F. DeMara, A. J. Gonzalez, M. Georgiopoulos, and H. Marshall. "PDU Bundling and Replication for Reduction of Distributed Simulation Communication Traffic". *Journal of Defense Modeling and Simulation*, 1:171–183, August 2004. URL `http://cal.ucf.edu/journal/j_vargas_demara_jdms_04.pdf`.

30. Vargas, J. J., R. F. DeMara, A. J. Gonzalez, and M. Georgiopoulos. "Bandwidth Analysis of a Simulated Computer Network Executing OTB". *Proceedings of the Second Swedish American Workshop on Modeling and Simulation (SAW-MAS'04)*, 201–208. Cocoa Beach, Florida, United States, February 2004. URL `http://www.mind.foi.se/SAWMAS/SAWMAS-2004/Papers/P18-SAWMAS-2004-J-Vargas.pdf`.

31. Wikipedia. "Image:Ethernet Type II Frame format.png — Wikipedia, The Free Encyclopedia", 2006. URL

http://en.wikipedia.org/w/index.php?title=Image:Ethernet_Type_II_Frame_format.png&oldid=16577459. [Online; accessed 19-May-2006].

32. Yuan, Zijing, Wentong Cai, Malcolm Y. H. Low, and Stephen J Turner. "Federate Migration in HLA-based Distributed Simulation". *Proceedings of the Workshop on HLA-based Distributed Simulation on the Grid*, volume 3038/2004, 856–864. Krakow, Poland, 2004.

33. Zabek, Anita A. "Lessons learned from the Design and Execution of a Federation for Joint Experimentation". *Proceedings of the 1999 Winter Simulation Conference*, volume 2, 1032–1038. Phoenix, Arizona, United States, 1999.

34. Zeng, Yi, Wentong Cai, and Stephen J. Turner. "Causal Order Based Time Warp: A Tradeoff of Optimism". *Proceedings of the 2003 Winter Simulation Conference*, 855–863. New Orleans, Louisiana, United States, 2003.

35. Zhou, Suiping, Stephen John Turner, Wentong Cai, Hanfeng Zhao, and Xiaolin Pang. "A Utility Model for Timely State Update in Distributed Wargame Simulations". *Proceedings of the 18th IEEE Workshop on Parallel and Distributed Simulation (PADS'04)*, 105–111. Kufstein, Austria, 2004.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704–0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202–4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
13–06–2006	Master's Thesis	Sept 2004 — Jun 2006

4. TITLE AND SUBTITLE

Managing Bandwidth and Traffic via Bundling and Filtration in Large-Scale Distributed Simulations

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Roberto C. Sánchez, 2d Lt, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
AFIT/EN Bldg. 641
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GCE/ENG/06-06

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Randyll Levine
(937) 255-0672
ASC/XRA
Modelling and Simulation Division
2180 8th Street
Wright-Patterson Air Force Base, OH 45433-7205

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approval for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Research has shown that bandwidth can be a limiting factor in the performance of distributed simulations. The Air Force's Distributed Mission Operations Center (DMOC) periodically hosts one of the largest distributed simulation events in the world. The engineers at the DMOC have dealt with the difficult problem of limited bandwidth by implementing application level filters that process all DIS PDUs between the various networks connected to the exercise. This thesis examines their implemented filter and proposes: adaptive range-based filtering and bundling together of PDUs. The goals are to reduce the number of PDUs passed by the adaptive filter and to reduce network overhead and the total amount of data transferred by maximizing packet size up to the MTU. The proposed changes were implemented and logged data from previous events were used on a test network in order to measure the improvement from the base filter to the improved filter. The results showed that the adaptive range based filter was effective, though minimally so, and that the PDU bundling resulted in a reduction of 17% to 20% of the total traffic transmitted across the network.

15. SUBJECT TERMS

distributed simulation, bandwidth utilization, network traffic management, bandwidth optimization, adaptive filtering

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Kenneth Hopkinson
U	U	U	UU	76	19b. TELEPHONE NUMBER *(include area code)* (937) 255–3636, ext 4579